```
EEEEEEEEEEEEEEE   RRRRRRRRRRR      FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
```

```
RRRRRRRR    EEEEEEEEEE    CCCCCCCC    SSSSSSSS  EEEEEEEEEE  LL          EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
RRRRRRRR    EEEEEEEEEE    CCCCCCCC    SSSSSSSS  EEEEEEEEEE  LL          EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
RR      RR  EE          CC          SS          EE          LL          EE          CC            TT
RR      RR  EE          CC          SS          EE          LL          EE          CC            TT
RR      RR  EE          CC          SS          EE          LL          EE          CC            TT
RR      RR  EE          CC          SS          EE          LL          EE          CC            TT
RRRRRRRR    EEEEEEEE    CC            SSSSSS    EEEEEEEE    LL          EEEEEEEE    CC            TT
RRRRRRRR    EEEEEEEE    CC            SSSSSS    EEEEEEEE    LL          EEEEEEEE    CC            TT
RR  RR      EE          CC                  SS  EE          LL          EE          CC            TT
RR  RR      EE          CC                  SS  EE          LL          EE          CC            TT
RR    RR    EE          CC                  SS  EE          LL          EE          CC            TT        ....
RR    RR    EE          CC                  SS  EE          LL          EE          CC            TT        ....
RR      RR  EEEEEEEEEE    CCCCCCCC  SSSSSSSS    EEEEEEEEEE  LLLLLLLLLL  EEEEEEEEEE    CCCCCCCC      TT        ....
RR      RR  EEEEEEEEEE    CCCCCCCC  SSSSSSSS    EEEEEEEEEE  LLLLLLLLLL  EEEEEEEEEE    CCCCCCCC      TT        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;  1          0001  0 MODULE RECSELECT
;  2          0002  0 (%TITLE 'Entry Validation'
;  3          0003  0 IDENT = 'V04-000') =
;  4          0004  0
;  5          0005  1 BEGIN
;  6          0006  1
;  7          0007  1 !
;  8          0008  1 !*****************************************************************************
;  9          0009  1 !*                                                                           *
; 10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
; 11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
; 12          0012  1 !*  ALL RIGHTS RESERVED.                                                    *
; 13          0013  1 !*                                                                         *
; 14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 19          0019  1 !*  TRANSFERRED.                                                            *
; 20          0020  1 !*                                                                         *
; 21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 23          0023  1 !*  CORPORATION.                                                            *
; 24          0024  1 !*                                                                         *
; 25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
; 27          0027  1 !*                                                                         *
; 28          0028  1 !*                                                                         *
; 29          0029  1 !*****************************************************************************
; 30          0030  1
; 31          0031  1 !++
; 32          0032  1 ! FACILITY:  ERF, Error Log Report Generator
; 33          0033  1 !
; 34          0034  1 ! ABSTRACT:
; 35          0035  1 !
; 36          0036  1 !       This routine will determine if the previously read entry
; 37          0037  1 !       meets user specified selection criteria.
; 38          0038  1 !
; 39          0039  1 ! ENVIRONMENT:
; 40          0040  1 !
; 41          0041  1 !       VAX/VMS operating system, user mode.
; 42          0042  1 !
; 43          0043  1 ! AUTHOR:  Sharon Reynolds,     CREATION DATE:  January 1983
; 44          0044  1 !
; 45          0045  1 ! Modified by:
; 46          0046  1 !
; 47          0047  1 !       V03-022 EAD0179         Elliott A. Drayton              6-Jul-1984
; 48          0048  1 !               Obtain LSTLUN value from SYECOM.
; 49          0049  1 !
; 50          0050  1 !       V03-023 SAR0274         Sharon A. Reynolds      19-Jun-1984
; 51          0051  1 !               - Added another check for device selection and entry
; 52          0052  1 !               selection combinations to fix a bug with
; 53          0053  1 !               /INC=(MF,VOLUME) and /INC=(TAPE,VOLUME).
; 54          0054  1 !
; 55          0055  1 !       V03-022 EAD0179         Elliott A. Drayton      23-May-1984
; 56          0056  1 !               Correct the passing of the address of device name
; 57          0057  1 !               in VERIFY_DEVICE.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-021 SAR0267         Sharon A. Reynolds      15-May-1984
;    60          0060  1 !               - Updated VERIFY_DEVICE to support longer device names.
;    61          0061  1 !               - Added check for unknown entry output to replace code
;    62          0062  1 !               that was previously removed.
;    63          0063  1 !
;    64          0064  1 !       V03-020 SAR0254         Sharon A. Reynolds      23-Apr-1984
;    65          0065  1 !               Added flag to /before check to stop execution when
;    66          0066  1 !               last entry found.
;    67          0067  1 !
;    68          0068  1 !       V03-019 EAD0151         Elliott A. Drayton      14-Apr-1984
;    69          0069  1 !               Fixed structure names in VERIFY_DEVICE.
;    70          0070  1 !
;    71          0071  1 !       V03-018 EAD0141         Elliott A. Drayton      12-Apr-1984
;    72          0072  1 !               Removed reference to EMBETDEF.
;    73          0073  1 !
;    74          0074  1 !       V03-017 SAR0248         Sharon A. Reynolds      10-Apr-1984
;    75          0075  1 !               Moved the unknown keyword tests to the verify entry
;    76          0076  1 !               routine so it would go through same tests as any
;    77          0077  1 !               other /include or /exclude entry selection.
;    78          0078  1 !
;    79          0079  1 !       V03-016 SAR0245         Sharon A. Reynolds      4-Apr-1984
;    80          0080  1 !               Added EMB$LOGMSP to device type entry table.
;    81          0081  1 !
;    82          0082  1 !       V03-015 EAD0119         Elliott A. Drayton      23-Mar-1984
;    83          0083  1 !               Remove support for /UNKNOWN qualifier and added support
;    84          0084  1 !               for the UNKNOWN keyword.
;    85          0085  1 !
;    86          0086  1 !       V03-014 EAD0115         Elliott A. Drayton       9-Mar-1984
;    87          0087  1 !               Removed emb_buf and syecom_buf.
;    88          0088  1 !
;    89          0089  1 !       V03-013 SAR0189         Sharon A. Reynolds,     13-Feb-1984
;    90          0090  1 !               - Added 'CS' device name support to device table search
;    91          0091  1 !               routine.
;    92          0092  1 !               - Added additional test for entry summary update.
;    93          0093  1 !
;    94          0094  1 !       V03-012 SAR0184         Sharon A. Reynolds,     17-Jan-1984
;    95          0095  1 !               - Fixed a bug in the output of the erf_unkentry message.
;    96          0096  1 !               - Added code to set the end value indicator when
;    97          0097  1 !                 the last selected entry (/entry) is found.
;    98          0098  1 !
;    99          0099  1 !       V03-011 SAR0181         Sharon A. Reynolds,     13-Dec-1983
;   100          0100  1 !               - Remove descriptor references.
;   101          0101  1 !               - Add device attention keyword support.
;   102          0102  1 !               - Add lm/sp entries to device errors entry list.
;   103          0103  1 !               - Add lm/sp entry check for bus class selections.
;   104          0104  1 !               - Removed logmessage keyword.
;   105          0105  1 !               - Add unsolicited_mscp keyword support.
;   106          0106  1 !               - Added incomplete entry message.
;   107          0107  1 !
;   108          0108  1 !       V03-010 SAR0176         Sharon A. Reynolds,     21-Nov-1983
;   109          0109  1 !               - Removed un-necessary check for outputting all
;   110          0110  1 !                 entries.
;   111          0111  1 !               - Changed reference to report type.
;   112          0112  1 !
;   113          0113  1 !       V03-009 SAR0152         Sharon A. Reynolds,     7-Oct-1983
;   114          0114  1 !               - Added code to output informational messages when
```

```
;  115     0115  1 !                 and unknown entry is encountered.
;  116     0116  1 !               - Added the code that counts intervening logmessage
;  117     0117  1 !                 logstatus entries.
;  118     0118  1 !               - Re-structured the /include and /exclude entry
;  119     0119  1 !                 checks to fix a bug.
;  120     0120  1 !               - Made /includ=disks/exclude=db1 a valid command.
;  121     0121  1 !
;  122     0122  1 !       V03-008 SAR0139         Sharon A. Reynolds,     20-Sep-1983
;  123     0123  1 !               Fixed a bug in mount/dismount output. Fixed an out
;  124     0124  1 !               of range loop.
;  125     0125  1 !
;  126     0126  1 !       V03-007 SAR0122         Sharon A. Reynolds,     23-Aug-1983
;  127     0127  1 !               Re-wrote translate_class routine for use with the
;  128     0128  1 !               permanent device tables.
;  129     0129  1 !
;  130     0130  1 !       V03-006 SAR0032         Sharon A. Reynolds,     2-Jun-1983
;  131     0131  1 !               Replaced emb_stuf with emb_buf definitions. Fixed bug
;  132     0132  1 !               in dc$_bus selection.
;  133     0133  1 !
;  134     0134  1 !       V03-005 SAR0029         Sharon A. Reynolds,     11-May-1983
;  135     0135  1 !               Removed support for logstatus keyword.
;  136     0136  1 !
;  137     0137  1 !       V03-004 SAR0013         Sharon A. Reynolds,     18-Apr-1983
;  138     0138  1 !               Deleted the log message and status message entries
;  139     0139  1 !               from the 'control' table. Added call to update
;  140     0140  1 !               entry summaries.
;  141     0141  1 !
;  142     0142  1 !       V03-003 SAR0003         Sharon A. Reynolds,     5-Apr-1983
;  143     0143  1 !               Removed the volume_output flag definition. Changed
;  144     0144  1 !               any references to volume_output flag so they refer
;  145     0145  1 !               to it from SYECOM.
;  146     0146  1 !
;  147     0147  1 !       V03-002 SAR0002         Sharon A. Reynolds,     5-Apr-1983
;  148     0148  1 !               Fixed /exclude selection bug.
;  149     0149  1 !
;  150     0150  1 !       V03-001 SAR0001         Sharon A. Reynolds,     29-Mar-1983
;  151     0151  1 !               Fixed /include='device name', volume mount/dismount
;  152     0152  1 !               selection problem.
;  153     0153  1 !
;  154     0154  1 !--
;  155     0155  1
;  156     0156  1
;  157     0157  1 !
;  158     0158  1 ! Required files
;  159     0159  1 !
;  160     0160  1 REQUIRE 'SRC$:ERFDEF.REQ' ;                 ! ERF defintions
;  161     0446  1 REQUIRE 'LIB$:PARSERDAT.R32' ;              ! ERF parser data definitions
;  162     0600  1 REQUIRE 'SRC$:RECSELDEF.REQ' ;              ! EMB, SYECOM, LOGMSG, LOGSTS, and
;  163     0731  1                                             !  VOLMOUNT field defintions
;  164     0732  1
;  165     0733  1 !
;  166     0734  1 ! Table of contents
;  167     0735  1 !
;  168     0736  1 !
;  169     0737  1 FORWARD ROUTINE
;  170     0738  1     Record_selected,                        ! Verify entry against selections
;  171     0739  1     Verify_entry,                           ! Verify the entry type
```

```
;  172          0740  1      Device_type_entry,                  ! Determine if it's a device type entry
;  173          0741  1      Verify_device_class,                ! Verify the device class
;  174          0742  1      Verify_device,                      ! Verify the device name
;  175          0743  1      Translate_class ;                   ! Translate device class to a name
;  176          0744  1
;  177          0745  1  !
;  178          0746  1  ! Declare external routines
;  179          0747  1  !
;  180          0748  1  EXTERNAL ROUTINE
;  181          0749  1      Exec_image,                         ! Execute an image
;  182          0750  1      Intervene_increment,
;  183          0751  1      Intervene_output,
;  184          0752  1      Search_queue: addressing_mode (general) , ! Search queue of devices selected
;  185          0753  1      Validate_packet;                    ! Is the packet validate for the cpu it was logged on.
;  186          0754  1
;  187          0755  1  !
;  188          0756  1  ! Declare external literals
;  189          0757  1  !
;  190          0758  1  EXTERNAL LITERAL
;  191          0759  1      Erf_incentry,
;  192          0760  1      Erf_unkclass,
;  193          0761  1      Erf_unkcpu,
;  194          0762  1      Erf_unkentry,
;  195          0763  1      Erf_unktype ;
;  196          0764  1
;  197          0765  1  !
;  198          0766  1  ! Declare external data.
;  199          0767  1  !
;  200          0768  1  EXTERNAL
;  201          0769  1      Class_dir:          REF $BBLOCK,
;  202          0770  1      Device_class,
;  203          0771  1      Device_type,
;  204          0772  1      Emb:                $BBLOCK PSECT (EMB),
;  205          0773  1      Exclude_flag,
;  206          0774  1      Exclude_mask:       REF $BBLOCK,
;  207          0775  1      Include_mask:       REF $BBLOCK,
;  208          0776  1      Option_flag:        REF $BBLOCK,
;  209          0777  1      Parser_data:        REF $BBLOCK,
;  210          0778  1      Processor_type,
;  211          0779  1      Summary_dispatcher_addr,
;  212          0780  1      Summary_flag:       REF $BBLOCK,
;  213          0781  1      Syecom:             $BBLOCK PSECT (SYECOM),
;  214          0782  1      Unknown_entry ;
;  215          0783  1
;  216          0784  1  !
;  217          0785  1  ! Declare literal definitions
;  218          0786  1  !
;  219          0787  1  LITERAL
;  220          0788  1      Incomplete_entry = 128 ;
;  221          0789  1
;  222          0790  1  !
;  223          0791  1  ! Own storage definitions
;  224          0792  1  !
;  225          0793  1  OWN
;  226          0794  1      Lstlun:             Long,
;  227          0795  1      Dev_selection_required: BYTE,
;  228          0796  1      Device_status:      BYTE,
```

```
;  229     0797  1     Dev_cls_status:     BYTE,
;  230     0798  1     Dev_type_entry_sts: BYTE,
;  231     0799  1     Entry_status:       BYTE,
;  232     0800  1     Validate_pkt_sts:   Initial (false),
;  233     0801  1     Bugchks:            VECTOR [3,byte,unsigned]  ! Bugcheck type entries
;  234     0802  1                         Initial (byte
;  235     0803  1                                 (EMB$K_CR,        ! Crash
;  236     0804  1                                  EMB$K_SBC,       ! System bugchecks
;  237     0805  1                                  EMB$K_UBC)),     ! User bugchecks
;  238     0806  1
;  239     0807  1     Control:            VECTOR [7,byte,unsigned] ! Control type entries
;  240     0808  1                         Initial (byte
;  241     0809  1                                 (EMB$K_CS,        ! Cold re-start
;  242     0810  1                                  EMB$K_NF,        ! New file created
;  243     0811  1                                  EMB$K_WS,        ! Warm re-start
;  244     0812  1                                  EMB$K_TS,        ! Time stamp
;  245     0813  1                                  EMB$K_SS,        ! System service message
;  246     0814  1                                  EMB$K_OM,        ! Operator message
;  247     0815  1                                  EMB$K_NM)),      ! Network message
;  248     0816  1
;  249     0817  1     Cpu:                VECTOR [8,byte,unsigned] ! Cpu type entries
;  250     0818  1                         Initial (byte
;  251     0819  1                                 (EMB$K_AW,        ! Asynchronouw write error
;  252     0820  1                                  EMB$K_UBA,       ! Unibus adapter error
;  253     0821  1                                  EMB$K_MBA,       ! Massbus adapter error
;  254     0822  1                                  EMB$K_UI,        ! Undefined interrupt
;  255     0823  1                                  EMB$K_BE,        ! Bus error
;  256     0824  1                                  EMB$K_SA,        ! SBI alert
;  257     0825  1                                  EMB$K_SI,        ! 11/750 fault thru SBI vector
;  258     0826  1                                  EMB$K_UE)),      ! 11/730 unibus error
;  259     0827  1
;  260     0828  1     Dev_errors:         VECTOR [3,byte,unsigned] ! Device error entries
;  261     0829  1                         Initial (byte
;  262     0830  1                                 (EMB$K_DE,        ! Device Errors
;  263     0831  1                                  EMB$K_SP,        ! Logstatus entries (mscp)
;  264     0832  1                                  EMB$K_LM)),      ! Logmessage entries (mscp)
;  265     0833  1
;  266     0834  1     Memorys:            VECTOR [2,byte,unsigned] ! Memory entries
;  267     0835  1                         Initial (byte
;  268     0836  1                                 (EMB$K_SE,        ! Soft ECC error
;  269     0837  1                                  EMB$K_HE)),      ! Hard ECC error
;  270     0838  1
;  271     0839  1     Volume:             VECTOR [2,byte,unsigned]        ! Volume change entries
;  272     0840  1                         Initial (byte
;  273     0841  1                                 (EMB$K_VM,        ! Volume mounts
;  274     0842  1                                  EMB$K_VD)) ;     ! Volume dismounts
;  275     0843  1
```

```
;  277     0844  1 GLOBAL ROUTINE RECORD_SELECTED =
;  278     0845  2 Begin
;  279     0846  2
;  280     0847  2 !++
;  281     0848  2 !
;  282     0849  2 ! Functional Description:
;  283     0850  2 !
;  284     0851  2 !       This routine will determine what selection qualifiers are
;  285     0852  2 !       specified and match the appropriate fields in the current
;  286     0853  2 !       entry against the selections. It return TRUE if the
;  287     0854  2 !       current entry matches or return FALSE if the current entry
;  288     0855  2 !       does NOT match.
;  289     0856  2 !
;  290     0857  2 ! Calling sequence:
;  291     0858  2 !
;  292     0859  2 !       RECORD_SELECTED ()
;  293     0860  2 !
;  294     0861  2 ! Input parameters:
;  295     0862  2 !
;  296     0863  2 !       None
;  297     0864  2 !
;  298     0865  2 ! Output parameters:
;  299     0866  2 !
;  300     0867  2 !       None
;  301     0868  2 !
;  302     0869  2 !--
;  303     0870  2
;  304     0871  2 LOCAL
;  305     0872  2     Include_status:     BYTE
;  306     0873  2                         Initial (true),
;  307     0874  2     Exclude_status:     BYTE
;  308     0875  2                         Initial (true) ;
;  309     0876  2
;  310     0877  2 lstlun = .syecom [sye$l_lstlun];
;  311     0878  2
;  312     0879  2 !
;  313     0880  2 ! Validate the packet for entry/cpu type and device class/type.
;  314     0881  2 !
;  315     0882  3 If NOT (VALIDATE_PACKET ())
;  316     0883  2 Then
;  317     0884  2     Unknown_entry = true
;  318     0885  2 Else
;  319     0886  2     Unknown_entry = false ;
;  320     0887  2
;  321     0888  2 !
;  322     0889  2 ! Determine if /summary selected and update that entry summary
;  323     0890  2 ! information.
;  324     0891  2 !
;  325     0892  3 If (.option_flag[opt$v_summary_qual] AND
;  326     0893  4     (.summary_flag[sum$v_entry] OR
;  327     0894  4      .summary_flag[sum$v_all_summ] OR
;  328     0895  3      .summary_flag[sum$v_histogram]))
;  329     0896  2 Then
;  330     0897  2     Exec_image (summary_dispatcher_addr,lstlun,%REF(entry_summ_upd)) ;
;  331     0898  2
;  332     0899  2 !
;  333     0900  2 ! If incomplete entry report the error.
```

```
;   334     0901  2 !
;   335     0902  3 If ((NOT .syecom[sye$b_valid_entry]) AND
;   336     0903  3     (.emb[emb$w_hd_entry] GEQ incomplete_entry))
;   337     0904  2 Then
;   338     0905  3     Begin
;   339     0906  3     Signal (erf_incentry, 1, .emb[emb$w_hd_entry]);
;   340     0907  3     Return false;
;   341     0908  2     End;
;   342     0909  2
;   343     0910  2 !
;   344     0911  2 ! Determine whether the volume mounts/dismounts should be output or just
;   345     0912  2 ! label information saved from the entry.
;   346     0913  2 !
;   347     0914  3 If (.exclude_mask[exc$v_volume] AND
;   348     0915  4     (.include_mask[inc$v_device_select] OR
;   349     0916  4     .include_mask[inc$v_dev_class_select] OR
;   350     0917  4     .include_mask[inc$v_dev_attentions] OR
;   351     0918  4     .include_mask[inc$v_dev_errors] OR
;   352     0919  2     .include_mask[inc$v_dev_timeouts])) AND
;   353     0920  3     (NOT .include_mask[inc$v_volume] OR
;   354     0921  3     NOT .option_flag[opt$v_output_all])
;   355     0922  2 Then
;   356     0923  2         !
;   357     0924  2         ! Indicate that volume mount/dismount entries
;   358     0925  2         ! should not be output.
;   359     0926  2         !
;   360     0927  2         Syecom[sye$b_volume_output] = false
;   361     0928  2 Else
;   362     0929  2         Syecom[sye$b_volume_output] = true ;
;   363     0930  2
;   364     0931  2 !
;   365     0932  2 ! Determine if the /ENTRY qualifier was specified.
;   366     0933  2 !
;   367     0934  2 If .option_flag[opt$v_entry_qual]
;   368     0935  2 Then
;   369     0936  2     !
;   370     0937  2     ! /Entry specified, get the address of the entry selection
;   371     0938  2     ! data and determine if the number of this entry
;   372     0939  2     ! is within the selected range.
;   373     0940  2     !
;   374     0941  3     Begin
;   375     0942  3     If .syecom[sye$l_reccnt] LSSU .parser_data[erl$l_end_entry]
;   376     0943  3     Then
;   377     0944  3         !
;   378     0945  3         ! This entry should be within the selected range, ensure
;   379     0946  3         ! the entry number is greater than the starting entry selection.
;   380     0947  3         !
;   381     0948  4         Begin
;   382     0949  5         If NOT (.syecom[sye$l_reccnt] GEQU .parser_data[erl$l_start_entry])
;   383     0950  4         Then
;   384     0951  4             !
;   385     0952  4             ! Entry is NOT within the selected range, return to calling
;   386     0953  4             ! routine.
;   387     0954  4             !
;   388     0955  4             Return false ;
;   389     0956  4         End
;   390     0957  3     Else
```

```
;  391     0958  3          !
;  392     0959  3          ! Entry is NOT within the selected range, return to calling
;  393     0960  3          ! routine.
;  394     0961  3          !
;  395     0962  4          Begin
;  396     0963  4          If .syecom[sye$l_reccnt] GEQU .parser_data[erl$l_end_entry]
;  397     0964  4          Then
;  398     0965  4              !
;  399     0966  4              ! Indicate that last selected entry was found.
;  400     0967  4              !
;  401     0968  4              Syecom[sye$b_end_value] = true ;
;  402     0969  4
;  403     0970  4          Return true ;
;  404     0971  3          End ;
;  405     0972  2      End ;
;  406     0973  2
;  407     0974  2  !
;  408     0975  2  ! Determine if the /BEFORE qualifier was specified.
;  409     0976  2  !
;  410     0977  2  If .option_flag[opt$v_before_qual]
;  411     0978  2  Then
;  412     0979  2      !
;  413     0980  2      ! Determine if the date/time that this entry was recorded falls
;  414     0981  2      ! within the range of selected date/times.
;  415     0982  2      !
;  416     0983  3      Begin
;  417   P 0984  3      If COMPARE_QUAD(emb[emb$q_hd_time],GEQU,
;  418     0985  4                      parser_data[erl$q_end_date])
;  419     0986  3      Then
;  420     0987  3          !
;  421     0988  3          ! This entry is NOT within the selected date/time range,
;  422     0989  3          ! return to the calling routine.
;  423     0990  3          !
;  424     0991  4          Begin
;  425     0992  4          Syecom[sye$b_end_value] = true ;
;  426     0993  4          Return true ;
;  427     0994  3          End ;
;  428     0995  2      End ;
;  429     0996  2
;  430     0997  2  !
;  431     0998  2  ! Determine if the /SINCE qualifier was specified.
;  432     0999  2  !
;  433     1000  2  If .option_flag[opt$v_since_qual]
;  434     1001  2  Then
;  435     1002  2      !
;  436     1003  2      ! Ensure that the entry date/time is greater than the starting
;  437     1004  2      ! time/date selection.
;  438     1005  2      !
;  439     1006  3      Begin
;  440   P 1007  3      If NOT COMPARE_QUAD(emb[emb$q_hd_time],GEQU,
;  441     1008  4                          parser_data[erl$q_start_date])
;  442     1009  3      Then
;  443     1010  3          !
;  444     1011  3          ! The entry does NOT meet that selection criteria for date/time,
;  445     1012  3          ! return to the calling routine.
;  446     1013  3          !
;  447     1014  3          Return false ;
```

```
;  448          1015  2     End ;
;  449          1016  2
;  450          1017  2 !
;  451          1018  2 ! Determine if the /SID_REGISTER qualifier was specified.
;  452          1019  2 !
;  453          1020  2 If .option_flag[opt$v_sid_reg_qual]
;  454          1021  2 Then
;  455          1022  2     !
;  456          1023  2     ! Determine if the entry sid matches the selected sid.
;  457          1024  2     !
;  458          1025  3     Begin
;  459          1026  3     If NOT .parser_data[erl$l_sid_selection] EQLU .emb[emb$l_hd_sid]
;  460          1027  3     Then
;  461          1028  3         !
;  462          1029  3         ! Entry sid does NOT match selected sid, return to calling
;  463          1030  3         ! routine.
;  464          1031  3         !
;  465          1032  3         Return false ;
;  466          1033  2     End ;
;  467          1034  2
;  468          1035  2 Device_status = false ;
;  469          1036  2 Dev_cls_status = false ;
;  470          1037  2 Entry_status = false ;
;  471          1038  2
;  472          1039  2 Dev_type_entry_sts = DEVICE_TYPE_ENTRY () ;
;  473          1040  2
;  474          1041  2 If .option_flag[opt$v_include_qual]
;  475          1042  2 Then
;  476          1043  3     Begin
;  477          1044  3     Exclude_flag = false ;
;  478          1045  3
;  479          1046  3     If .dev_type_entry_sts OR
;  480          1047  3         (.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  481          1048  4         (.emb[emb$w_hd_entry] EQLU EMB$K_VD)
;  482          1049  3     Then
;  483          1050  4         Begin
;  484          1051  4         If .include_mask[inc$v_device_select]
;  485          1052  4         Then
;  486          1053  5             Begin
;  487          1054  5             If VERIFY_DEVICE ()
;  488          1055  5             Then
;  489          1056  5                 Device_status = true
;  490          1057  5             Else
;  491          1058  5                 Device_status = false ;
;  492          1059  4             End ;
;  493          1060  4
;  494          1061  4         If .include_mask[inc$v_dev_class_select]
;  495          1062  4         Then
;  496          1063  5             Begin
;  497          1064  5             If VERIFY_DEVICE_CLASS ()
;  498          1065  5             Then
;  499          1066  5                 Dev_cls_status = true
;  500          1067  5             Else
;  501          1068  5                 Dev_cls_status = false ;
;  502          1069  4             End ;
;  503          1070  3         End ;
;  504          1071  3
```

```
;  505          1072  3     If .include_mask[inc$v_entry_select]
;  506          1073  3     Then
;  507          1074  4         Begin
;  508          1075  4         If VERIFY_ENTRY ()
;  509          1076  4         Then
;  510          1077  4             Entry_status = true
;  511          1078  4         Else
;  512          1079  4             Entry_status = false ;
;  513          1080  3         End ;
;  514          1081  3
;  515          1082  3
;  516          1083  4     If (.include_mask[inc$v_device_select] AND
;  517          1084  3         .dev_type_entry_sts AND .device_status) OR
;  518          1085  3
;  519          1086  4         (.include_mask[inc$v_dev_class_select] AND
;  520          1087  3         .dev_type_entry_sts AND .dev_class_status) OR
;  521          1088  3
;  522          1089  4         (.include_mask[inc$v_entry_select] AND .entry_status)
;  523          1090  3     Then
;  524          1091  3         Include_status = true
;  525          1092  3     Else
;  526          1093  3         Include_status = false ;
;  527          1094  3
;  528          1095  3
;  529          1096  3     If .include_mask[inc$v_device_select] AND
;  530          1097  3         .include_mask[inc$v_entry_select]
;  531          1098  3     Then
;  532          1099  4         Begin
;  533          1100  4         Include_status = false ;
;  534          1101  4
;  535          1102  4         If .dev_selection_required
;  536          1103  4         Then
;  537          1104  5             Begin
;  538          1105  5             If (.entry_status AND .device_status) OR
;  539          1106  6                 (.dev_type_entry_sts AND .device_status)
;  540          1107  5             Then
;  541          1108  5                 Include_status = true ;
;  542          1109  5             End
;  543          1110  4         Else
;  544          1111  5             Begin
;  545          1112  5             If .dev_type_entry_sts AND .device_status
;  546          1113  5             Then
;  547          1114  6                 Begin
;  548          1115  6                 Include_status = true ;
;  549          1116  6                 End
;  550          1117  5             Else
;  551          1118  6                 Begin
;  552          1119  7                 If (NOT .dev_type_entry_sts AND .entry_status)
;  553          1120  6                 Then
;  554          1121  6                     Include_status = true ;
;  555          1122  5                 End ;
;  556          1123  4             End ;
;  557          1124  3         End ;
;  558          1125  3
;  559          1126  3     If .include_mask[inc$v_dev_class_select] AND
;  560          1127  3         .include_mask[inc$v_entry_select]
;  561          1128  3     Then
```

```
;  562     1129  4          Begin
;  563     1130  4          Include_status = false ;
;  564     1131  4
;  565     1132  4          If .dev_selection_required
;  566     1133  4          Then
;  567     1134  5              Begin
;  568     1135  5              If (.entry_status AND .dev_cls_status) OR
;  569     1136  6                  (.dev_type_entry_sts AND .dev_cls_status)
;  570     1137  5              Then
;  571     1138  5                  Include_status = true ;
;  572     1139  5              End
;  573     1140  4          Else
;  574     1141  5              Begin
;  575     1142  5              If .dev_type_entry_sts AND .dev_cls_status
;  576     1143  5              Then
;  577     1144  6                  Begin
;  578     1145  6                  Include_status = true ;
;  579     1146  6                  End
;  580     1147  5              Else
;  581     1148  6                  Begin
;  582     1149  7                  If (NOT .dev_type_entry_sts AND .entry_status)
;  583     1150  6                  Then
;  584     1151  6                      Include_status = true ;
;  585     1152  5                  End ;
;  586     1153  4              End ;
;  587     1154  3          End ;
;  588     1155  3
;  589     1156  2      End ;
;  590     1157  2
;  591     1158  2 !
;  592     1159  2 !If not /include option then include_status = false
;  593     1160  2 !
;  594     1161  2
;  595     1162  2 If .option_flag[opt$v_exclude_qual]
;  596     1163  2 Then
;  597     1164  3     Begin
;  598     1165  3     Exclude_flag = true ;
;  599     1166  3
;  600     1167  3     If .dev_type_entry_sts OR
;  601     1168  3         (.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  602     1169  4         (.emb[emb$w_hd_entry] EQLU EMB$K_VD)
;  603     1170  3     Then
;  604     1171  4         Begin
;  605     1172  4         If .exclude_mask[exc$v_device_select]
;  606     1173  4         Then
;  607     1174  5             Begin
;  608     1175  5             If VERIFY_DEVICE ()
;  609     1176  5             Then
;  610     1177  5                 Device_status = true
;  611     1178  5             Else
;  612     1179  5                 Device_status = false ;
;  613     1180  4             End ;
;  614     1181  4
;  615     1182  4         If .exclude_mask[exc$v_dev_class_select]
;  616     1183  4         Then
;  617     1184  5             Begin
;  618     1185  5             If VERIFY_DEVICE_CLASS ()
```

```
;   619          1186  5              Then
;   620          1187  5                  Dev_cls_status = true
;   621          1188  5              Else
;   622          1189  5                  Dev_cls_status = false ;
;   623          1190  4              End ;
;   624          1191  3          End ;
;   625          1192  3
;   626          1193  3      If .exclude_mask[exc$v_entry_select]
;   627          1194  3      Then
;   628          1195  4          Begin
;   629          1196  4          If VERIFY_ENTRY ()
;   630          1197  4          Then
;   631          1198  4              Entry_status = true
;   632          1199  4          Else
;   633          1200  4              Entry_status = false ;
;   634          1201  3          End ;
;   635          1202  3
;   636          1203  4      If (.exclude_mask[exc$v_device_select] AND
;   637          1204  3          .dev_type_entry_sts AND .device_status) OR
;   638          1205  3
;   639          1206  4          (.exclude_mask[exc$v_dev_class_select] AND
;   640          1207  3          .dev_type_entry_sts AND .dev_cls_status) OR
;   641          1208  3
;   642          1209  4          (.exclude_mask[exc$v_entry_select] AND .entry_status)
;   643          1210  3      Then
;   644          1211  3          Exclude_status = false
;   645          1212  3      Else
;   646          1213  3          Exclude_status = true ;
;   647          1214  3
;   648          1215  3
;   649          1216  3      If .exclude_mask[exc$v_device_select] AND
;   650          1217  3          .exclude_mask[exc$v_entry_select]
;   651          1218  3      Then
;   652          1219  4          Begin
;   653          1220  4          Exclude_status = true ;
;   654          1221  4
;   655          1222  4          If .dev_selection_required
;   656          1223  4          Then
;   657          1224  5              Begin
;   658          1225  5              If (.entry_status AND .device_status) OR
;   659          1226  6                  (.dev_type_entry_sts AND .device_status)
;   660          1227  5              Then
;   661          1228  5                  Exclude_status = false ;
;   662          1229  5              End
;   663          1230  4          Else
;   664          1231  5              Begin
;   665          1232  5              If .dev_type_entry_sts AND .device_status
;   666          1233  5              Then
;   667          1234  6                  Begin
;   668          1235  6                  Exclude_status = false ;
;   669          1236  6                  End
;   670          1237  5              Else
;   671          1238  6                  Begin
;   672          1239  7                  If (NOT .dev_type_entry_sts AND .entry_status)
;   673          1240  6                  Then
;   674          1241  6                      Exclude_status = false ;
;   675          1242  5                  End ;
```

```
;  676     1243  4                End ;
;  677     1244  3            End ;
;  678     1245  3
;  679     1246  3        If .exclude_mask[exc$v_dev_class_select] AND
;  680     1247  3           .exclude_mask[exc$v_entry_select]
;  681     1248  3        Then
;  682     1249  4            Begin
;  683     1250  4            Exclude_status = true ;
;  684     1251  4
;  685     1252  4            If .dev_selection_required
;  686     1253  4            Then
;  687     1254  5                Begin
;  688     1255  5                If (.entry_status AND .dev_cls_status) OR
;  689     1256  6                    (.dev_type_entry_sts AND .dev_cls_status)
;  690     1257  5                Then
;  691     1258  5                    Exclude_status = false ;
;  692     1259  5                End
;  693     1260  4            Else
;  694     1261  5                Begin
;  695     1262  5                If .dev_type_entry_sts AND .dev_cls_status
;  696     1263  5                Then
;  697     1264  6                    Begin
;  698     1265  6                    Exclude_status = false ;
;  699     1266  6                    End
;  700     1267  5                Else
;  701     1268  6                    Begin
;  702     1269  7                    If (NOT .dev_type_entry_sts AND .entry_status)
;  703     1270  6                    Then
;  704     1271  6                        Exclude_status = false ;
;  705     1272  5                    End ;
;  706     1273  4                End ;
;  707     1274  3            End ;
;  708     1275  3
;  709     1276  2        End ;        ! of /exclude processing
;  710     1277  2    !
;  711     1278  2    ! IF /exclude option match, exclude_status = false.
;  712     1279  2    !
;  713     1280  2
;  714     1281  2
;  715     1282  2    !
;  716     1283  2    ! Determine whether to count logmessage/logstatus entries.
;  717     1284  2    !
;  718     1285  3    If ( (.include_status) AND (.exclude_status) AND
;  719     1286  3        (.parser_data[erl$b_rpt_type] EQ[ full_rep) )
;  720     1287  2    Then
;  721     1288  2        !
;  722     1289  2        ! Determine if it was a logmessage/logstatus entry.
;  723     1290  2        !
;  724     1291  3        Begin
;  725     1292  3        If (.emb[emb$w_hd_entry] EQLU EMB$C_SP) OR
;  726     1293  4            (.emb[emb$w_hd_entry] EQLU EMB$C_LM)
;  727     1294  3        Then
;  728     1295  3            !
;  729     1296  3            ! Count the number of logmessage/logstatus entries
;  730     1297  3            ! that might be skipped.
;  731     1298  3            !
;  732     1299  3            INTERVENE_INCREMENT (lstlun)
```

```
;   733          1300  3     Else
;   734          1301  3         !
;   735          1302  3         ! Determine whether to output the logstatus/logmessage
;   736          1303  3         ! intervening message and if necessary output it.
;   737          1304  3         !
;   738          1305  3         INTERVENE_OUTPUT (lstlun) ;
;   739          1306  2     End ;
;   740          1307  2
;   741          1308  2 !
;   742          1309  2 ! Determine if the entry met the selection criteria.
;   743          1310  2 !
;   744          1311  2 ! Determine if this is an unknown entry.
;   745          1312  2 !
;   746          1313  2 If .unknown_entry
;   747          1314  2 Then
;   748          1315  2     ! Indicate that this is an unknown entry and return with a
;   749          1316  2     ! true value so that it will be output.
;   750          1317  2     !
;   751          1318  2     Return true ;
;   752          1319  2
;   753          1320  2 If (NOT .include_status) OR
;   754          1321  3     (NOT .exclude_status)
;   755          1322  2 Then
;   756          1323  2     !
;   757          1324  2     ! Indicate that the entry should not be output by
;   758          1325  2     ! returning to the calling routine with a false value.
;   759          1326  2     !
;   760          1327  2     Return false ;
;   761          1328  2
;   762          1329  2 !
;   763          1330  2 ! Indicate that the entry should be output by
;   764          1331  2 ! returning to the calling routine with a true value.
;   765          1332  2 !
;   766          1333  2 Return true ;
;   767          1334  2
;   768          1335  1 End ;   ! Routine


                                                .TITLE  RECSELECT Entry Validation
                                                .IDENT  \V04-000\

                                                .PSECT  $OWN$,NOEXE,  PIC,2

                                    00000 LSTLUN:   .BLKB   4
                                    00004 DEV_SELECTION_REQUIRED:
                                                    .BLKB   1
                                    00005 DEVICE_STATUS:
                                                    .BLKB   1
                                    00006 DEV_CLS_STATUS:
                                                    .BLKB   1
                                    00007 DEV_TYPE_ENTRY_STS:
                                                    .BLKB   1
                                    00008 ENTRY_STATUS:
                                                    .BLKB   1
                                    00009           .BLKB   3
                          00000000  0000C VALIDATE_PKT_STS:
                                                    .LONG   0                                       ;
```

```
                                                70  28  25  00010 BUGCHKS:.BYTE   37, 40, 112                                    ;
                                                                00013         .BLKB   1
                                            2A  29  27  26  24  23  20  00014 CONTROL:.BYTE   32, 35, 36, 38, 39, 41, 42         ;
                                                                0001B         .BLKB   1
                                        0B  0A  05  04  61  0C  09  07  0001C CPU:     .BYTE   7, 9, 12, 97, 4, 5, 10, 11         ;
                                                64  63  01  00024 DEV_ERRORS:
                                                                              .BYTE   1, 99, 100                                 ;
                                                                00027         .BLKB   1
                                                    08  06  00028 MEMORYS:.BYTE   6, 8                                         ;
                                                                0002A         .BLKB   2
                                                    41  40  0002C VOLUME: .BYTE   64, 65                                       ;

                                                                              .EXTRN  EXEC_IMAGE, INTERVENE_INCREMENT
                                                                              .EXTRN  INTERVENE_OUTPUT
                                                                              .EXTRN  SEARCH_QUEUE, VALIDATE_PACKET
                                                                              .EXTRN  ERF_INCENTRY, ERF_UNKCLASS
                                                                              .EXTRN  ERF_UNKCPU, ERF_UNKENTRY
                                                                              .EXTRN  ERF_UNKTYPE, CLASS_DIR
                                                                              .EXTRN  DEVICE_CLASS, DEVICE_TYPE
                                                                              .EXTRN  EMB, EXCLUDE_FLAG
                                                                              .EXTRN  EXCLUDE_MASK, INCLUDE_MASK
                                                                              .EXTRN  OPTION_FLAG, PARSER_DATA
                                                                              .EXTRN  PROCESSOR_TYPE, SUMMARY_DISPATCHER_ADDR
                                                                              .EXTRN  SUMMARY_FLAG, SYECOM
                                                                              .EXTRN  UNKNOWN_ENTRY

                                                                              .PSECT  $CODE,NOWRT,  PIC,2

                                                        0FFC 00000            .ENTRY  RECORD_SELECTED, Save R2,R3,R4,R5,R6,R7,R8,-; 0844
                                                                                      R9,R10,R11
                                                5B  00000000G   00  9E 00002          MOVAB   PARSER_DATA, R11                   ;
                                                5A  00000000G   00  9E 00009          MOVAB   EXCLUDE_MASK, R10                  ;
                                                59  00000000G   00  9E 00010          MOVAB   OPTION_FLAG, R9                    ;
                                                58  00000000G   00  9E 00017          MOVAB   SYECOM+24, R8                      ;
                                                57  00000000G   00  9E 0001E          MOVAB   INCLUDE_MASK, R7                   ;
                                                56  00000000G   00  9E 00025          MOVAB   EMB+4, R6                          ;
                                                55  00000000'   00  9E 0002C          MOVAB   ENTRY_STATUS, R5                   ;
                                                5E              04  C2 00033          SUBL2   #4, SP                             ;
                                                54              01  90 00036          MOVB    #1, INCLUDE_STATUS                 ; 0845
                                                53              01  90 00039          MOVB    #1, EXCLUDE_STATUS                 ;
                                            F8  A5      0F      A8  D0 0003C          MOVL    SYECOM+39, LSTLUN                  ; 0877
                                00000000G   00              00  FB 00041          CALLS   #0, VALIDATE_PACKET                ; 0882
                                            09              50  E8 00048          BLBS    R0, 1$                             ;
                                00000000G   00              01  D0 0004B          MOVL    #1, UNKNOWN_ENTRY                  ; 0884
                                                            06  11 00052          BRB     2$                                 ;
                                                00000000G   00  D4 00054 1$:      CLRL    UNKNOWN_ENTRY                      ; 0886
                                                50          69  D0 0005A 2$:      MOVL    OPTION_FLAG, R0                    ; 0892
                                27              60          0E  E1 0005D          BBC     #14, (R0), 4$                      ;
                                                50  00000000G   00  D0 00061      MOVL    SUMMARY_FLAG, R0                   ; 0893
                                07              60          02  E0 00068          BBS     #2, (R0), 3$                       ;
                                                04          60  E8 0006C          BLBS    (R0), 3$                           ; 0894
                                15              60          05  E1 0006F          BBC     #5, (R0), 4$                       ; 0895
                                                6E          05  D0 00073 3$:      MOVL    #5, (SP)                           ; 0897
                                                            5E  DD 00076          PUSHL   SP                                 ;
                                                        F8  A5  9F 00078          PUSHAB  LSTLUN                             ;
                                                00000000G   00  9F 0007B          PUSHAB  SUMMARY_DISPATCHER_ADDR            ;
                                00000000G   00              03  FB 00081          CALLS   #3, EXEC_IMAGE                     ;
```

```
                    1C        03    A8  E8 00088 4$:     BLBS    SYECOM+27, 6$                   ; 0902
              0080  8F              66  B1 0008C         CMPW    EMB+4, #128                     ; 0903
                                    15  1F 00091         BLSSU   6$
                    7E              66  3C 00093         MOVZWL  EMB+4, -(SP)                    ; 0906
                                    01  DD 00096         PUSHL   #1
                    00000000G       8F  DD 00098         PUSHL   #ERF_INCENTRY
         00000000G  00              03  FB 0009E         CALLS   #3, LIB$SIGNAL
                              02E9  31  000A5 5$:        BRW     69$                             ; 0907
                    50              6A  D0 000A8 6$:     MOVL    EXCLUDE_MASK, R0                ; 0914
              29    60              12  E1 000AB         BBC     #18, (R0), 9$
                    50              67  D0 000AF         MOVL    INCLUDE_MASK, R0                ; 0915
              10    60              14  E0 000B2         BBS     #20, (R0), 7$
              0C    60              15  E0 000B6         BBS     #21, (R0), 7$                   ; 0916
              08    60              09  E0 000BA         BBS     #9, (R0), 7$                    ; 0917
              04    60              0D  E0 000BE         BBS     #13, (R0), 7$                   ; 0918
                    12        02    A0  E9 000C2         BLBC    2(R0), 9$                       ; 0919
                    50              67  D0 000C6 7$:     MOVL    INCLUDE_MASK, R0                ; 0920
              07    60              12  E1 000C9         BBC     #18, (R0), 8$
                    50              69  D0 000CD         MOVL    OPTION_FLAG, R0                 ; 0921
                                    60  B5 000D0         TSTW    (R0)
                                    04  19 000D2         BLSS    9$
                                    68  94 000D4 8$:     CLRB    SYECOM+24                       ; 0927
                                    03  11 000D6         BRB     10$
                    68              01  90 000D8 9$:     MOVB    #1, SYECOM+24                   ; 0929
                    52              69  D0 000DB 10$:    MOVL    OPTION_FLAG, R2                 ; 0934
              13    62              03  E1 000DE         BBC     #3, (R2), 11$
                    51        E8    A8  D0 000E2         MOVL    SYECOM, R1                      ; 0942
                    50              6B  D0 000E6         MOVL    PARSER_DATA, R0
                    19        A0    51  D1 000E9         CMPL    R1, 25(R0)
                                    1D  1E 000ED         BGEQU   13$
                    15        A0    51  D1 000EF         CMPL    R1, 21(R0)                      ; 0949
                                    B0  1F 000F3         BLSSU   5$
                    1B              62  E9 000F5 11$:    BLBC    (R2), 14$                       ; 0977
              50    6B              05  C1 000F8         ADDL3   #5, PARSER_DATA, R0             ; 0985
                    51        06    A6  D0 000FC         MOVL    A+4, R1
                    04        A0    51  D1 00100         CMPL    R1, 4(R0)
                                    04  12 00104         BNEQ    12$
                    60        02    A6  D1 00106         CMPL    A, (R0)
                                    07  1F 0010A 12$:     BLSSU   14$
                    06        A8    01  90 0010C 13$:     MOVB    #1, SYECOM+30                   ; 0992
                              027A  31  00110            BRW     68$                             ; 0993
              18    62              0D  E1 00113 14$:     BBC     #13, (R2), 16$                  ; 1000
              50    6B              0D  C1 00117         ADDL3   #13, PARSER_DATA, R0            ; 1008
                    51        06    A6  D0 0011B         MOVL    A+4, R1
                    04        A0    51  D1 0011F         CMPL    R1, 4(R0)
                                    08  12 00123         BNEQ    15$
                    60        02    A6  D1 00125         CMPL    A, (R0)
                                    12  1F 00129         BLSSU   17$
                                    02  11 0012B         BRB     16$
                                    0E  1F 0012D 15$:     BLSSU   17$
              0D    62              0C  E1 0012F 16$:     BBC     #12, (R2), 18$                  ; 1020
                    50              6B  D0 00133         MOVL    PARSER_DATA, R0                 ; 1026
                    FC        A6    01  A0 D1 00136      CMPL    1(R0), EMB
                                    03  13 0013B         BEQL    18$
                              0251  31  0013D 17$:     BRW     69$
                          FD  A5    B4  00140 18$:     CLRW    DEVICE_STATUS                   ; 1035
                                    65  94 00143         CLRB    ENTRY_STATUS                    ; 1037
```

```
                00000000V  00              00  FB  00145          CALLS   #0, DEVICE_TYPE_ENTRY                   ; 1039
                      FF   A5              50  90  0014C          MOVB    R0, DEV_TYPE_ENTRY_STS
                           50              69  D0  00150          MOVL    OPTION_FLAG, R0                        ; 1041
                03         60              06  E0  00153          BBS     #6, (R0), 19$
                                        00F3  31  00157          BRW     41$
                                00000000G  00  D4  0015A 19$:      CLRL    EXCLUDE_FLAG                           ; 1044
                           0E       FF     A5  E8  00160          BLBS    DEV_TYPE_ENTRY_STS, 20$                ; 1046
                      0040 8F               66  B1  00164          CMPW    EMB+4, #64                             ; 1047
                                            07  13  00169          BEQL    20$
                      0041 8F               66  B1  0016B          CMPW    EMB+4, #65                             ; 1048
                                            34  12  00170          BNEQ    24$
                           50               67  D0  00172 20$:      MOVL    INCLUDE_MASK, R0                       ; 1051
                13         60              14  E1  00175          BBC     #20, (R0), 22$
                00000000V  00              00  FB  00179          CALLS   #0, VERIFY_DEVICE                      ; 1054
                           06              50  E9  00180          BLBC    R0, 21$
                      FD   A5              01  90  00183          MOVB    #1, DEVICE_STATUS                      ; 1056
                                            03  11  00187          BRB     22$
                                      FD    A5  94  00189 21$:      CLRB    DEVICE_STATUS                          ; 1058
                           50               67  D0  0018C 22$:      MOVL    INCLUDE_MASK, R0                       ; 1061
                13         60              15  E1  0018F          BBC     #21, (R0), 24$
                00000000V  00              00  FB  00193          CALLS   #0, VERIFY_DEVICE_CLASS                ; 1064
                           06              50  E9  0019A          BLBC    R0, 23$
                      FE   A5              01  90  0019D          MOVB    #1, DEV_CLS_STATUS                     ; 1066
                                            03  11  001A1          BRB     24$
                                      FE    A5  94  001A3 23$:      CLRB    DEV_CLS_STATUS                         ; 1068
                           50               67  D0  001A6 24$:      MOVL    INCLUDE_MASK, R0                       ; 1072
                11         60              16  E1  001A9          BBC     #22, (R0), 26$
                00000000V  00              00  FB  001AD          CALLS   #0, VERIFY_ENTRY                       ; 1075
                           05              50  E9  001B4          BLBC    R0, 25$
                           65              01  90  001B7          MOVB    #1, ENTRY_STATUS                       ; 1077
                                            02  11  001BA          BRB     26$
                                            65  94  001BC 25$:      CLRB    ENTRY_STATUS                           ; 1079
                           50               67  D0  001BE 26$:      MOVL    INCLUDE_MASK, R0                       ; 1083
                08         60              14  E1  001C1          BBC     #20, (R0), 27$
                           04       FF     A5  E9  001C5          BLBC    DEV_TYPE_ENTRY_STS, 27$                ; 1084
                           13       FD     A5  E8  001C9          BLBS    DEVICE_STATUS, 29$
                08         60              15  E1  001CD 27$:      BBC     #21, (R0), 28$                         ; 1086
                           04       FF     A5  E9  001D1          BLBC    DEV_TYPE_ENTRY_STS, 28$                ; 1087
                           07       FE     A5  E8  001D5          BLBS    DEV_CLS_STATUS, 29$
                08         60              16  E1  001D9 28$:      BBC     #22, (R0), 30$                         ; 1089
                           05              65  E9  001DD          BLBC    ENTRY_STATUS, 30$
                           54              01  90  001E0 29$:      MOVB    #1, INCLUDE_STATUS                     ; 1091
                                            02  11  001E3          BRB     31$
                                            54  94  001E5 30$:      CLRB    INCLUDE_STATUS                         ; 1093
                2F         60              14  E1  001E7 31$:      BBC     #20, (R0), 36$                         ; 1096
                2B         60              16  E1  001EB          BBC     #22, (R0), 36$                         ; 1097
                                            54  94  001EF          CLRB    INCLUDE_STATUS                         ; 1100
                           11       FC     A5  E9  001F1          BLBC    DEV_SELECTION_REQUIRED, 33$            ; 1102
                           04       65     E9  001F5              BLBC    ENTRY_STATUS, 32$                      ; 1105
                           1B       FD     A5  E8  001F8          BLBS    DEVICE_STATUS, 35$
                           1A       FF     A5  E9  001FC 32$:      BLBC    DEV_TYPE_ENTRY_STS, 36$                ; 1106
                           16       FD     A5  E9  00200          BLBC    DEVICE_STATUS, 36$
                                            11  11  00204          BRB     35$                                    ; 1108
                           51       FF     A5  9A  00206 33$:      MOVZBL  DEV_TYPE_ENTRY_STS, R1                 ; 1112
                           07              51  E9  0020A          BLBC    R1, 34$
                           06       FD     A5  E8  0020D          BLBS    DEVICE_STATUS, 35$
                           06              51  E8  00211          BLBS    R1, 36$                                ; 1119
```

```
                        03              65  E9  00214 34$:     BLBC    ENTRY_STATUS, 36$                                       ;
                        54              01  90  00217 35$:     MOVB    #1, INCLUDE_STATUS                                      ; 1121
                    2F  60              15  E1  0021A 36$:     BBC     #21, (R0), 41$                                          ; 1126
                    2B  60              16  E1  0021E          BBC     #22, (R0), 41$                                          ; 1127
                                        54  94  00222          CLRB    INCLUDE_STATUS                                          ; 1130
                        11          FC  A5  E9  00224          BLBC    DEV_SELECTION_REQUIRED, 38$                             ; 1132
                        04              65  E9  00228          BLBC    ENTRY_STATUS, 37$                                       ; 1135
                        1B          FE  A5  E8  0022B          BLBS    DEV_CLS_STATUS, 40$                                     ;
                        1A          FF  A5  E9  0022F 37$:     BLBC    DEV_TYPE_ENTRY_STS, 41$                                 ; 1136
                        16          FE  A5  E9  00233          BLBC    DEV_CLS_STATUS, 41$                                     ;
                                        11  11  00237          BRB     40$                                                     ; 1138
                        50          FF  A5  9A  00239 38$:     MOVZBL  DEV_TYPE_ENTRY_STS, R0                                  ; 1142
                        07              50  E9  0023D          BLBC    R0, 39$                                                 ;
                        06          FE  A5  E8  00240          BLBS    DEV_CLS_STATUS, 40$                                     ;
                        06              50  E8  00244          BLBS    R0, 41$                                                 ; 1149
                        03              65  E9  00247 39$:     BLBC    ENTRY_STATUS, 41$                                       ;
                        54              01  90  0024A 40$:     MOVB    #1, INCLUDE_STATUS                                      ; 1151
                        50              69  D0  0024D 41$:     MOVL    OPTION_FLAG, R0                                         ; 1162
                    03  60              04  E0  00250          BBS     #4, (R0), 42$                                           ;
                                    00F4  31  00254          BRW     64$                                                     ;
            00000000G   00              01  D0  00257 42$:     MOVL    #1, EXCLUDE_FLAG                                        ; 1165
                        0E          FF  A5  E8  0025E          BLBS    DEV_TYPE_ENTRY_STS, 43$                                 ; 1167
                  0040  8F              66  B1  00262          CMPW    EMB+4, #64                                              ; 1168
                                        07  13  00267          BEQL    43$                                                     ;
                  0041  8F              66  B1  00269          CMPW    EMB+4, #65                                              ; 1169
                                        34  12  0026E          BNEQ    47$                                                     ;
                        50              6A  D0  00270 43$:     MOVL    EXCLUDE_MASK, R0                                        ; 1172
                    13  60              14  E1  00273          BBC     #20, (R0), 45$                                          ;
            00000000V   00              00  FB  00277          CALLS   #0, VERIFY_DEVICE                                       ; 1175
                        06              50  E9  0027E          BLBC    R0, 44$                                                 ;
                    FD  A5              01  90  00281          MOVB    #1, DEVICE_STATUS                                       ; 1177
                                        03  11  00285          BRB     45$                                                     ;
                        FD  A5          94  00287 44$:         CLRB    DEVICE_STATUS                                           ; 1179
                        50              6A  D0  0028A 45$:     MOVL    EXCLUDE_MASK, R0                                        ; 1182
                    13  60              15  E1  0028D          BBC     #21, (R0), 47$                                          ;
            00000000V   00              00  FB  00291          CALLS   #0, VERIFY_DEVICE_CLASS                                 ; 1185
                        06              50  E9  00298          BLBC    R0, 46$                                                 ;
                    FE  A5              01  90  0029B          MOVB    #1, DEV_CLS_STATUS                                      ; 1187
                                        03  11  0029F          BRB     47$                                                     ;
                        FE  A5          94  002A1 46$:         CLRB    DEV_CLS_STATUS                                          ; 1189
                        50              6A  D0  002A4 47$:     MOVL    EXCLUDE_MASK, R0                                        ; 1193
                    11  60              16  E1  002A7          BBC     #22, (R0), 49$                                          ;
            00000000V   00              00  FB  002AB          CALLS   #0, VERIFY_ENTRY                                        ; 1196
                        05              50  E9  002B2          BLBC    R0, 48$                                                 ;
                        65              01  90  002B5          MOVB    #1, ENTRY_STATUS                                        ; 1198
                                        02  11  002B8          BRB     49$                                                     ;
                                        65  94  002BA 48$:     CLRB    ENTRY_STATUS                                            ; 1200
                        50              6A  D0  002BC 49$:     MOVL    EXCLUDE_MASK, R0                                        ; 1203
                    08  60              14  E1  002BF          BBC     #20, (R0), 50$                                          ;
                        04          FF  A5  E9  002C3          BLBC    DEV_TYPE_ENTRY_STS, 50$                                 ; 1204
                        13          FD  A5  E8  002C7          BLBS    DEVICE_STATUS, 52$                                      ;
                    08  60              15  E1  002CB 50$:     BBC     #21, (R0), 51$                                          ; 1206
                        04          FF  A5  E9  002CF          BLBC    DEV_TYPE_ENTRY_STS, 51$                                 ; 1207
                        07          FE  A5  E8  002D3          BLBS    DEV_CLS_STATUS, 52$                                     ;
                    07  60              16  E1  002D7 51$:     BBC     #22, (R0), 53$                                          ; 1209
                        04              65  E9  002DB          BLBC    ENTRY_STATUS, 53$                                       ;
                                        53  94  002DE 52$:     CLRB    EXCLUDE_STATUS                                          ; 1211
```

```
                               03  11 002E0       BRB     54$                                   ;
                      53       01  90 002E2 53$:  MOVB    #1, EXCLUDE_STATUS                    ; 1213
          2F          60       14  E1 002E5 54$:  BBC     #20, (R0), 59$                        ; 1216
          2B          60       16  E1 002E9       BBC     #22, (R0), 59$                        ; 1217
                      53       01  90 002ED       MOVB    #1, EXCLUDE_STATUS                    ; 1220
                      11  FC   A5  E9 002F0       BLBC    DEV_SELECTION_REQUIRED, 56$           ; 1222
                      04       65  E9 002F4       BLBC    ENTRY_STATUS, 55$                     ; 1225
                      1B  FD   A5  E8 002F7       BLBS    DEVICE_STATUS, 58$                    ;
                      19  FF   A5  E9 002FB 55$:  BLBC    DEV_TYPE_ENTRY_STS, 59$               ; 1226
                      15  FD   A5  E9 002FF       BLBC    DEVICE_STATUS, 59$                    ;
                               11  11 00303       BRB     58$                                   ; 1228
                      51  FF   A5  9A 00305 56$:  MOVZBL  DEV_TYPE_ENTRY_STS, R1                ; 1232
                      07       51  E9 00309       BLBC    R1, 57$                               ;
                      06  FD   A5  E8 0030C       BLBS    DEVICE_STATUS, 58$                    ;
                      05       51  E8 00310       BLBS    R1, 59$                               ; 1239
                      02       65  E9 00313 57$:  BLBC    ENTRY_STATUS, 59$                     ;
                      53       94 00316 58$:      CLRB    EXCLUDE_STATUS                        ; 1241
          2F          60       15  E1 00318 59$:  BBC     #21, (R0), 64$                        ; 1246
          2B          60       16  E1 0031C       BBC     #22, (R0), 64$                        ; 1247
                      53       01  90 00320       MOVB    #1, EXCLUDE_STATUS                    ; 1250
                      11  FC   A5  E9 00323       BLBC    DEV_SELECTION_REQUIRED, 61$           ; 1252
                      04       65  E9 00327       BLBC    ENTRY_STATUS, 60$                     ; 1255
                      1B  FE   A5  E8 0032A       BLBS    DEV_CLS_STATUS, 63$                   ;
                      19  FF   A5  E9 0032E 60$:  BLBC    DEV_TYPE_ENTRY_STS, 64$               ; 1256
                      15  FE   A5  E9 00332       BLBC    DEV_CLS_STATUS, 64$                   ;
                               11  11 00336       BRB     63$                                   ; 1258
                      50  FF   A5  9A 00338 61$:  MOVZBL  DEV_TYPE_ENTRY_STS, R0                ; 1262
                      07       50  E9 0033C       BLBC    R0, 62$                               ;
                      06  FE   A5  E8 0033F       BLBS    DEV_CLS_STATUS, 63$                   ;
                      05       50  E8 00343       BLBS    R0, 64$                               ; 1269
                      02       65  E9 00346 62$:  BLBC    ENTRY_STATUS, 64$                     ;
                      53       94 00349 63$:      CLRB    EXCLUDE_STATUS                        ; 1271
                      32       54  E9 0034B 64$:  BLBC    INCLUDE_STATUS, 67$                   ; 1285
                      2F       53  E9 0034E       BLBC    EXCLUDE_STATUS, 67$                   ;
                      50       6B  D0 00351       MOVL    PARSER_DATA, R0                       ; 1286
                      02       60  91 00354       CMPB    (R0), #2                              ;
                               27  12 00357       BNEQ    67$                                   ;
                      50       66  3C 00359       MOVZWL  EMB+4, R0                             ; 1292
              0063    8F       50  B1 0035C       CMPW    R0, #99                               ;
                               07  13 00361       BEQL    65$                                   ;
              0064    8F       50  B1 00363       CMPW    R0, #100                              ; 1293
                               0C  12 00368       BNEQ    66$                                   ;
                          F8   A5  9F 0036A 65$:  PUSHAB  LSTLUN                                ; 1299
          00000000G   00       01  FB 0036D       CALLS   #1, INTERVENE_INCREMENT               ;
                               0A  11 00374       BRB     67$                                   ;
                          F8   A5  9F 00376 66$:  PUSHAB  LSTLUN                                ; 1305
          00000000G   00       01  FB 00379       CALLS   #1, INTERVENE_OUTPUT                  ;
                      06 00000000G 00  E8 00380 67$:  BLBS    UNKNOWN_ENTRY, 68$                ; 1313
                      07       54  E9 00387       BLBC    INCLUDE_STATUS, 69$                   ; 1320
                      04       53  E9 0038A       BLBC    EXCLUDE_STATUS, 69$                   ; 1321
                      50       01  D0 0038D 68$:  MOVL    #1, R0                                ; 1333
                               04 00390           RET                                           ;
                      50       D4 00391 69$:      CLRL    R0                                    ; 1335
                               04 00393           RET                                           ;

; Routine Size:  916 bytes,    Routine Base:  $CODE + 0000
```

```
;  769          1336  1
;  770          1337  1
```

```
;  772     1338  1 ROUTINE VERIFY_ENTRY =
;  773     1339  2 Begin
;  774     1340  2
;  775     1341  2 !++
;  776     1342  2 !
;  777     1343  2 ! Functional Description:
;  778     1344  2 !
;  779     1345  2 !       This routine will determine if the current entry matches
;  780     1346  2 !       any of the selected entry types. It return TRUE if the
;  781     1347  2 !       current entry matches or return FALSE if the current entry
;  782     1348  2 !       does NOT match.
;  783     1349  2 !
;  784     1350  2 ! Calling sequence:
;  785     1351  2 !
;  786     1352  2 !       VERIFY_ENTRY ()
;  787     1353  2 !
;  788     1354  2 ! Input parameters:
;  789     1355  2 !
;  790     1356  2 !       None
;  791     1357  2 !
;  792     1358  2 ! Output parameters:
;  793     1359  2 !
;  794     1360  2 !       None
;  795     1361  2 !
;  796     1362  2 !--
;  797     1363  2
;  798     1364  2
;  799     1365  2 !
;  800     1366  2 ! Initialize a status indicator.
;  801     1367  2 !
;  802     1368  2 Dev_selection_required = false ;
;  803     1369  2
;  804     1370  2 !
;  805     1371  2 ! Determine if device attention entries are selected.
;  806     1372  2 !
;  807     1373  3 If ((.exclude_mask[exc$v_dev_attentions]) OR
;  808     1374  3     (.include_mask[inc$v_dev_attentions]))
;  809     1375  2 Then
;  810     1376  2     !
;  811     1377  2     ! Determine if this entry is for a device attention.
;  812     1378  2     !
;  813     1379  3     Begin
;  814     1380  3     Dev_selection_required = true ;
;  815     1381  3     If .emb[emb$w_hd_entry] EQLU EMB$K_DA
;  816     1382  3     Then
;  817     1383  3         !
;  818     1384  3         ! Indicate that this entry does match a selected entry
;  819     1385  3         ! type, by returning to the calling routine with a
;  820     1386  3         ! true value.
;  821     1387  3         !
;  822     1388  3         Return true ;
;  823     1389  2     End ;
;  824     1390  2
;  825     1391  2 !
;  826     1392  2 ! Determine if bugcheck entries are selected.
;  827     1393  2 !
;  828     1394  3 If ((.exclude_mask[exc$v_bugchks]) OR
```

```
;   829          1395  3     (.include_mask[inc$v_bugchks]))
;   830          1396  2 Then
;   831          1397  2     !
;   832          1398  2     ! Determine if this entry is for a bugcheck.
;   833          1399  2     !
;   834          1400  3     Begin
;   835          1401  3     Incr I from 0 to 2 do
;   836          1402  4         Begin
;   837          1403  4         If .emb[emb$w_hd_entry] EQLU .bugchks[.I]
;   838          1404  4         Then
;   839          1405  4             !
;   840          1406  4             ! Indicate that this entry does match a selected
;   841          1407  4             ! entry type, by returning to the calling routine
;   842          1408  4             ! with a true value.
;   843          1409  4             !
;   844          1410  4             Return true ;
;   845          1411  3         End ;
;   846          1412  2     End ;
;   847          1413  2
;   848          1414  2 !
;   849          1415  2 ! Determine if 'control entries' are selected.
;   850          1416  2 !
;   851          1417  3 If ((.exclude_mask[exc$v_control_entry]) OR
;   852          1418  3     (.include_mask[inc$v_control_entry]))
;   853          1419  2 Then
;   854          1420  2     !
;   855          1421  2     ! Determine if this entry is a 'control entry'.
;   856          1422  2     !
;   857          1423  3     Begin
;   858          1424  3     Incr I from 0 to 6 do
;   859          1425  4         Begin
;   860          1426  4         If .emb[emb$w_hd_entry] EQLU .control[.I]
;   861          1427  4         Then
;   862          1428  4             !
;   863          1429  4             ! Indicate that this entry does match a selected
;   864          1430  4             ! entry type, by returning to the calling routine
;   865          1431  4             ! with a true value.
;   866          1432  4             !
;   867          1433  4             Return true ;
;   868          1434  3         End ;
;   869          1435  2     End ;
;   870          1436  2
;   871          1437  2 !
;   872          1438  2 ! Determine if 'cpu entries' are selected.
;   873          1439  2 !
;   874          1440  3 If ((.exclude_mask[exc$v_cpu_entry]) OR
;   875          1441  3     (.include_mask[inc$v_cpu_entry]))
;   876          1442  2 Then
;   877          1443  2     !
;   878          1444  2     ! Determine if this entry is a 'cpu entry'.
;   879          1445  2     !
;   880          1446  3     Begin
;   881          1447  3     Incr I from 0 to 7 do
;   882          1448  4         Begin
;   883          1449  4         If .emb[emb$w_hd_entry] EQLU .cpu[.I]
;   884          1450  4         Then
;   885          1451  4             !
```

```
;  886     1452  4                ! Indicate that this entry does match a selected
;  887     1453  4                ! entry type, by returning to the calling routine
;  888     1454  4                ! with a true value.
;  889     1455  4                !
;  890     1456  4                Return true ;
;  891     1457  3            End ;
;  892     1458  2        End ;
;  893     1459  2
;  894     1460  2    !
;  895     1461  2    ! Determine if device errors are selected.
;  896     1462  2    !
;  897     1463  3    If ((.exclude_mask[exc$v_dev_errors]) OR
;  898     1464  3        (.include_mask[inc$v_dev_errors]))
;  899     1465  2    Then
;  900     1466  2        !
;  901     1467  2        ! Determine if this entry is a device error.
;  902     1468  2        !
;  903     1469  3        Begin
;  904     1470  3        Dev_selection_required = true ;
;  905     1471  3
;  906     1472  3        Incr I from 0 to 2 do
;  907     1473  4            Begin
;  908     1474  4            If .emb[emb$w_hd_entry] EQLU .dev_errors[.I]
;  909     1475  4            Then
;  910     1476  4                !
;  911     1477  4                ! Indicate that this entry does match a selected
;  912     1478  4                ! entry type, by returning to the calling routine
;  913     1479  4                ! with a true value.
;  914     1480  4                !
;  915     1481  4                Return true ;
;  916     1482  3            End ;
;  917     1483  2        End ;
;  918     1484  2
;  919     1485  2    !
;  920     1486  2    ! Determine if machine checks are selected.
;  921     1487  2    !
;  922     1488  3    If ((.exclude_mask[exc$v_machine_chks]) OR
;  923     1489  3        (.include_mask[inc$v_machine_chks]))
;  924     1490  2    Then
;  925     1491  2        !
;  926     1492  2        ! Determine if this entry is a machine check.
;  927     1493  2        !
;  928     1494  3        Begin
;  929     1495  3        If .emb[emb$w_hd_entry] EQLU EMB$K_MC
;  930     1496  3        Then
;  931     1497  3            !
;  932     1498  3            ! Indicate that this entry does match a selected
;  933     1499  3            ! entry type, by returning to the calling routine
;  934     1500  3            ! with a true value.
;  935     1501  3            !
;  936     1502  3            Return true ;
;  937     1503  2        End ;
;  938     1504  2
;  939     1505  2    !
;  940     1506  2    ! Determine if memory entries are selected.
;  941     1507  2    !
;  942     1508  3    If ((.exclude_mask[exc$v_memory]) OR
```

```
;  943          1509  3     (.include_mask[inc$v_memory]))
;  944          1510  2 Then
;  945          1511  2     !
;  946          1512  2     ! Determine if this entry is a 'memory entry'.
;  947          1513  2     !
;  948          1514  3     Begin
;  949          1515  3     Incr I from 0 to 1 do
;  950          1516  4         Begin
;  951          1517  4         If .emb[emb$w_hd_entry] EQLU .memorys[.I]
;  952          1518  4         Then
;  953          1519  4             !
;  954          1520  4             ! Indicate that this entry does match a selected
;  955          1521  4             ! entry type, by returning to the calling routine
;  956          1522  4             ! with a true value.
;  957          1523  4             !
;  958          1524  4             Return true ;
;  959          1525  3         End ;
;  960          1526  2     End ;
;  961          1527  2
;  962          1528  2 !
;  963          1529  2 ! Determine if device timeouts are selected.
;  964          1530  2 !
;  965          1531  3 If ((.exclude_mask[exc$v_dev_timeouts]) OR
;  966          1532  3     (.include_mask[inc$v_dev_timeouts]))
;  967          1533  2 Then
;  968          1534  2     !
;  969          1535  2     ! Determine if this entry is a device timeouts.
;  970          1536  2     !
;  971          1537  3     Begin
;  972          1538  3     Dev_selection_required = true ;
;  973          1539  3
;  974          1540  3     If .emb[emb$w_hd_entry] EQLU EMB$K_DT
;  975          1541  3     Then
;  976          1542  3         !
;  977          1543  3         ! Indicate that this entry does match a selected
;  978          1544  3         ! entry type, by returning to the calling routine
;  979          1545  3         ! with a true value.
;  980          1546  3         !
;  981          1547  3         Return true ;
;  982          1548  2     End ;
;  983          1549  2
;  984          1550  2
;  985          1551  2 !
;  986          1552  2 ! Determine if unknown entries have been selected.
;  987          1553  2 ! If unknown entries have not been excluded, then see if this is an
;  988          1554  2 ! unknown entry. If it is set UNKNOWN_ENTRY true.
;  989          1555  2 !
;  990          1556  2 ! Initialize the unknown entry indicator (not an unknown entry).
;  991          1557  2 !
;  992          1558  3 If ((.exclude_mask[exc$v_unknown_entry]) OR
;  993          1559  3     (.include_mask[inc$v_unknown_entry]))
;  994          1560  2 Then
;  995          1561  2     !
;  996          1562  2     ! Determine if this is an unknown entry.
;  997          1563  2     !
;  998          1564  3     Begin
;  999          1565  3     If .unknown_entry
```

```
;  1000          1566  3     Then  Return true ;
;  1001          1567  2     End ;
;  1002          1568  2
;  1003          1569  2 !
;  1004          1570  2 ! Determine if unsolicited mscp entries are selected.
;  1005          1571  2 !
;  1006          1572  3 If ((.exclude_mask[exc$v_unsol_mscp]) OR
;  1007          1573  3     (.include_mask[inc$v_unsol_mscp]))
;  1008          1574  2 Then
;  1009          1575  2     !
;  1010          1576  2     ! Determine if this entry is an unsolicited mscp entry.
;  1011          1577  2     !
;  1012          1578  3     Begin
;  1013          1579  3     If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP
;  1014          1580  3     Then
;  1015          1581  3         !
;  1016          1582  3         ! Indicate that this entry does match a selected
;  1017          1583  3         ! entry type, by returning to the calling routine
;  1018          1584  3         ! with a true value.
;  1019          1585  3         !
;  1020          1586  3         Return true ;
;  1021          1587  2     End ;
;  1022          1588  2
;  1023          1589  2 !
;  1024          1590  2 ! Determine if volume changes are to be excluded.
;  1025          1591  2 !
;  1026          1592  4 If ((.exclude_mask[exc$v_volume])
;  1027          1593  3     OR (.include_mask[inc$v_volume]))
;  1028          1594  2 Then
;  1029          1595  2     !
;  1030          1596  2     ! Determine if this entry is a volume entry.
;  1031          1597  2     !
;  1032          1598  3     Begin
;  1033          1599  3     Dev_selection_required = true ;
;  1034          1600  3
;  1035          1601  3     Incr I from 0 to 1 do
;  1036          1602  4         Begin
;  1037          1603  4         If .emb[emb$w_hd_entry] EQLU .volume[.I]
;  1038          1604  4         Then
;  1039          1605  4             !
;  1040          1606  4             ! Indicate that this entry does match a selected
;  1041          1607  4             ! entry type, by returning to the calling routine
;  1042          1608  4             ! with a true value.
;  1043          1609  4             !
;  1044          1610  4             Return true ;
;  1045          1611  3         End ;
;  1046          1612  2     End ;
;  1047          1613  2
;  1048          1614  2 !
;  1049          1615  2 ! Indicate that this entry does not match any of the selected
;  1050          1616  2 ! entry types, by returning to the calling routine with a
;  1051          1617  2 ! false value.
;  1052          1618  2 !
;  1053          1619  2 Return false ;
;  1054          1620  1 End ;   ! Routine
```

```
                                003C 00000 VERIFY_ENTRY:
                                           .WORD   Save R2,R3,R4,R5                        ; 1338
                55 00000000G   00    9E 00002          MOVAB   EMB+4, R5
                54 00000000'   00    9E 00009          MOVAB   DEV_SELECTION_REQUIRED, R4
                53 00000000G   00    9E 00010          MOVAB   INCLUDE_MASK, R3
                               64    94 00017          CLRB    DEV_SELECTION_REQUIRED          ; 1368
                51 00000000G   00    D0 00019          MOVL    EXCLUDE_MASK, R1                ; 1373
          07    61             09    E0 00020          BBS     #9, (R1), 1$
                50             63    D0 00024          MOVL    INCLUDE_MASK, R0                ; 1374
          0A    60             09    E1 00027          BBC     #9, (R0), 2$
                64             01    90 0002B 1$:      MOVB    #1, DEV_SELECTION_REQUIRED      ; 1380
      0062      8F             65    B1 0002E          CMPW    EMB+4, #98                      ; 1381
                               7D    13 00033          BEQL    16$
          07    61             0A    E0 00035 2$:      BBS     #10, (R1), 3$                   ; 1394
                50             63    D0 00039          MOVL    INCLUDE_MASK, R0                ; 1395
          10    60             0A    E1 0003C          BBC     #10, (R0), 5$
                               50    D4 00040 3$:      CLRL    I                               ; 1403
                52       0C A440    9A 00042 4$:      MOVZBL  BUGCHKS[I], R2
                65             52    B1 00047          CMPW    R2, EMB+4
                               7D    13 0004A          BEQL    20$
          F2    50             02    F3 0004C          AOBLEQ  #2, I, 4$                       ; 1401
          07    61             0B    E0 00050 5$:      BBS     #11, (R1), 6$                   ; 1417
                50             63    D0 00054          MOVL    INCLUDE_MASK, R0                ; 1418
          10    60             0B    E1 00057          BBC     #11, (R0), 8$
                               50    D4 0005B 6$:      CLRL    I                               ; 1426
                52       10 A440    9A 0005D 7$:      MOVZBL  CONTROL[I], R2
                65             52    B1 00062          CMPW    R2, EMB+4
                               7B    13 00065          BEQL    23$
          F2    50             06    F3 00067          AOBLEQ  #6, I, 7$                       ; 1424
          07    61             0C    E0 0006B 8$:      BBS     #12, (R1), 9$                   ; 1440
                50             63    D0 0006F          MOVL    INCLUDE_MASK, R0                ; 1441
          10    60             0C    E1 00072          BBC     #12, (R0), 11$
                               50    D4 00076 9$:      CLRL    I                               ; 1449
                52       18 A440    9A 00078 10$:     MOVZBL  CPU[I], R2
                65             52    B1 0007D          CMPW    R2, EMB+4
                               60    13 00080          BEQL    23$
          F2    50             07    F3 00082          AOBLEQ  #7, I, 10$                      ; 1447
          07    61             0D    E0 00086 11$:     BBS     #13, (R1), 12$                  ; 1463
                50             63    D0 0008A          MOVL    INCLUDE_MASK, R0                ; 1464
          13    60             0D    E1 0008D          BBC     #13, (R0), 14$
                64             01    90 00091 12$:     MOVB    #1, DEV_SELECTION_REQUIRED      ; 1470
                               50    D4 00094          CLRL    I                               ; 1474
                52       20 A440    9A 00096 13$:     MOVZBL  DEV_ERRORS[I], R2
                65             52    B1 0009B          CMPW    R2, EMB+4
                               66    13 0009E          BEQL    28$
          F2    50             02    F3 000A0          AOBLEQ  #2, I, 13$                      ; 1472
          07    61             0E    E0 000A4 14$:     BBS     #14, (R1), 15$                  ; 1488
                50             63    D0 000A8          MOVL    INCLUDE_MASK, R0                ; 1489
          05    60             0E    E1 000AB          BBC     #14, (R0), 17$
                02             65    B1 000AF 15$:     CMPW    EMB+4, #2                       ; 1495
                               6E    13 000B2 16$:     BEQL    32$
                               61    B5 000B4 17$:     TSTW    (R1)                            ; 1508
                               07    19 000B6          BLSS    18$
                50             63    D0 000B8          MOVL    INCLUDE_MASK, R0                ; 1509
```

```
                                        60  B5 000BB          TSTW    (R0)                                    ;
                                        10  18 000BD          BGEQ    21$                                     ;
                                        50  D4 000BF 18$:     CLRL    I                                       ; 1517
                                52  24 A440  9A 000C1 19$:     MOVZBL  MEMORYS[I], R2                          ;
                                65       52  B1 000C6          CMPW    R2, EMB+4                               ;
                                        57  13 000C9 20$:     BEQL    32$                                     ;
                            F2      50       01  F3 000CB          AOBLEQ  #1, I, 19$                      ; 1515
                                07      02   A1  E8 000CF 21$:     BLBS    2(R1), 22$                      ; 1531
                                50           63  D0 000D3          MOVL    INCLUDE_MASK, R0                ; 1532
                                0A      02   A0  E9 000D6          BLBC    2(R0), 24$                      ;
                                64           01  90 000DA 22$:     MOVB    #1, DEV_SELECTION_REQUIRED      ; 1538
                        0060    8F           65  B1 000DD          CMPW    EMB+4, #96                      ; 1540
                                            3E  13 000E2 23$:     BEQL    32$                             ;
                            07      61       13  E0 000E4 24$:     BBS     #19, (R1), 25$                  ; 1558
                                50           63  D0 000E8          MOVL    INCLUDE_MASK, R0                ; 1559
                            07      60       13  E1 000EB          BBC     #19, (R0), 26$                  ;
                                2C  00000000G    00  E8 000EF 25$:     BLBS    UNKNOWN_ENTRY, 32$              ; 1565
                            07      61       11  E0 000F6 26$:     BBS     #17, (R1), 27$                  ; 1572
                                50           63  D0 000FA          MOVL    INCLUDE_MASK, R0                ; 1573
                            07      60       11  E1 000FD          BBC     #17, (R0), 29$                  ;
                        0065    8F           65  B1 00101 27$:     CMPW    EMB+4, #101                     ; 1579
                                            1A  13 00106 28$:     BEQL    32$                             ;
                            07      61       12  E0 00108 29$:     BBS     #18, (R1), 30$                  ; 1592
                                50           63  D0 0010C          MOVL    INCLUDE_MASK, R0                ; 1593
                            17      60       12  E1 0010F          BBC     #18, (R0), 34$                  ;
                                64           01  90 00113 30$:     MOVB    #1, DEV_SELECTION_REQUIRED      ; 1599
                                50           D4  00116          CLRL    I                               ; 1603
                                51  28 A440  9A 00118 31$:     MOVZBL  VOLUME[I], R1                           ;
                                65       51  B1 0011D          CMPW    R1, EMB+4                               ;
                                        04  12 00120          BNEQ    33$                                     ;
                                50           01  D0 00122 32$:     MOVL    #1, R0                          ; 1610
                                            04  00125          RET                                             ;
                            EE  50           01  F3 00126 33$:     AOBLEQ  #1, I, 31$                      ; 1601
                                50           D4  00129 34$:     CLRL    R0                              ; 1619
                                            04  0012C          RET                                             ; 1620

; Routine Size:  301 bytes,    Routine Base:  $CODE + 0394

; 1055          1621  1
```

```
;  1057          1622  1 GLOBAL ROUTINE DEVICE_TYPE_ENTRY =
;  1058          1623  2 Begin
;  1059          1624  2
;  1060          1625  2 !++
;  1061          1626  2 !
;  1062          1627  2 ! Functional Description:
;  1063          1628  2 !
;  1064          1629  2 !       This routine will determine if the current entry is a device
;  1065          1630  2 !       type entry; (device attention, device error, device timeout,
;  1066          1631  2 !       volume dismount, volume mount). It return TRUE if the current
;  1067          1632  2 !       entry matches any of the device type entries or return FALSE
;  1068          1633  2 !       if the current entry does NOT match.
;  1069          1634  2 !
;  1070          1635  2 ! Calling sequence:
;  1071          1636  2 !
;  1072          1637  2 !       DEVICE_ENTRY_TYPE ()
;  1073          1638  2 !
;  1074          1639  2 ! Input parameters:
;  1075          1640  2 !
;  1076          1641  2 !       None
;  1077          1642  2 !
;  1078          1643  2 ! Output parameters:
;  1079          1644  2 !
;  1080          1645  2 !       None
;  1081          1646  2 !
;  1082          1647  2 !--
;  1083          1648  2
;  1084          1649  2 OWN
;  1085          1650  2     Device_entries:     VECTOR [6,byte,unsigned] ! Storage for device type
;  1086          1651  2                                                ! entries.
;  1087          1652  2                         Initial (BYTE
;  1088          1653  2                                 (EMB$K_DA,     ! Device attentions
;  1089          1654  2                                 EMB$K_DE,      ! Device errors
;  1090          1655  2                                 EMB$K_DT,      ! Device timeouts
;  1091          1656  2                                 EMB$K_LM,
;  1092          1657  2                                 EMB$K_SP,      ! Log message
;  1093          1658  2                                 EMB$K_LOGMSCP)) ;! Unsolicited mscp msg
;  1094          1659  2
;  1095          1660  2 !
;  1096          1661  2 ! Determine if the current entry is a device type entry.
;  1097          1662  2 !
;  1098          1663  2 Incr I from 0 to 5 do
;  1099          1664  3     Begin
;  1100          1665  3     If .emb[emb$w_hd_entry] EQLU .device_entries[.I]
;  1101          1666  3     Then
;  1102          1667  3         !
;  1103          1668  3         ! Indicate that this is a device type entry, by
;  1104          1669  3         ! returning to the calling routine with a true value.
;  1105          1670  3         !
;  1106          1671  3         Return true ;
;  1107          1672  2     End ;
;  1108          1673  2
;  1109          1674  2 !
;  1110          1675  2 ! Indicate that this is NOT a device type entry, by returning
;  1111          1676  2 ! to the calling routine with a false value.
;  1112          1677  2 !
;  1113          1678  2 Return false ;
```

```
; 1114          1679  2
; 1115          1680  1 End ;   ! Routine


                                                        .PSECT  $OWN$,NOEXE,  PIC,2

                                                 0002E  .BLKB   2
                          65  63  64  60  01  62 00030 DEVICE_ENTRIES:
                                                        .BYTE   98, 1, 96, 100, 99, 101                 ;


                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                            0000 00000          .ENTRY  DEVICE_TYPE_ENTRY, Save nothing         ; 1622
                                         50   D4 00002          CLRL    I                                       ; 1665
                       51 00000000'0040   9A 00004 1$:       MOVZBL  DEVICE_ENTRIES[I], R1                   ;
                00000000G  00            51   B1 0000C          CMPW    R1, EMB+4                               ;
                                         04   12 00013          BNEQ    2$                                      ;
                           50            01   D0 00015          MOVL    #1, R0                                  ; 1671
                                            04 00018          RET                                             ;
                      E7   50            05   F3 00019 2$:     AOBLEQ  #5, I, 1$                               ; 1663
                                         50   D4 0001D          CLRL    R0                                      ; 1678
                                            04 0001F          RET                                             ; 1680

; Routine Size:  32 bytes,    Routine Base:  $CODE + 04C1

; 1116          1681  1
```

```
;  1118          1682  1 ROUTINE VERIFY_DEVICE_CLASS =
;  1119          1683  2 Begin
;  1120          1684  2
;  1121          1685  2 !++
;  1122          1686  2 !
;  1123          1687  2 ! Functional Description:
;  1124          1688  2 !
;  1125          1689  2 !       This routine will determine if the device recorded by the
;  1126          1690  2 !       current entry matches any of the selected device class(es).
;  1127          1691  2 !       It return TRUE if the current entry matches or return FALSE
;  1128          1692  2 !       if the current entry does NOT match.
;  1129          1693  2 !
;  1130          1694  2 ! Calling sequence:
;  1131          1695  2 !
;  1132          1696  2 !       VERIFY_DEVICE_CLASS ()
;  1133          1697  2 !
;  1134          1698  2 ! Input parameters:
;  1135          1699  2 !
;  1136          1700  2 !       None
;  1137          1701  2 !
;  1138          1702  2 ! Output parameters:
;  1139          1703  2 !
;  1140          1704  2 !       None
;  1141          1705  2 !
;  1142          1706  2 !--
;  1143          1707  2
;  1144          1708  2 !
;  1145          1709  2 ! Determine whether this is a unsolicited mscp entry and
;  1146          1710  2 ! whether to continue.
;  1147          1711  2 !
;  1148          1712  2 If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
;  1149          1713  2     NOT .include_mask[inc$v_disks] AND
;  1150          1714  2      NOT .include_mask[inc$v_tapes]
;  1151          1715  2 Then
;  1152          1716  2     Return false ;
;  1153          1717  2
;  1154          1718  2 !
;  1155          1719  2 ! Determine if 'BUS' entries are selected.
;  1156          1720  2 !
;  1157          1721  3 If ((.exclude_mask[exc$v_buses]) OR
;  1158          1722  3     (.include_mask[inc$v_buses]))
;  1159          1723  2 Then
;  1160          1724  2     !
;  1161          1725  2     ! Determine if the device recorded by this entry, matches the
;  1162          1726  2     ! selected device class.
;  1163          1727  2     !
;  1164          1728  3     Begin
;  1165          1729  5     If  ((.emb[emb$w_hd_entry] EQLU EMB$K_LM AND
;  1166          1730  3         .emb[emb$b_lm_class] EQLU DC$_BUS)) OR
;  1167          1731  3
;  1168          1732  5         ((.emb[emb$w_hd_entry] EQLU EMB$K_SP AND
;  1169          1733  3         .emb[emb$b_sp_class] EQLU DC$_BUS)) OR
;  1170          1734  3
;  1171          1735  4         (.emb[emb$b_dv_class] EQLU DC$_BUS)
;  1172          1736  3     Then
;  1173          1737  3         !
;  1174          1738  3         ! Indicate that this entry does match a selected device
```

```
;  1175     1739  3          ! class, by returning to the calling routine with a
;  1176     1740  3          ! true value.
;  1177     1741  3          !
;  1178     1742  3          Return true ;
;  1179     1743  2      End ;
;  1180     1744  2
;  1181     1745  2  !
;  1182     1746  2  ! Determine if 'DISK' entries are selected.
;  1183     1747  2  !
;  1184     1748  3  if ((.exclude_mask[exc$v_disks]) OR
;  1185     1749  3      (.include_mask[inc$v_disks]))
;  1186     1750  2  Then
;  1187     1751  2      !
;  1188     1752  2      ! Determine if the device recorded by this entry, matches the
;  1189     1753  2      ! selected device class.
;  1190     1754  2      !
;  1191     1755  3      Begin
;  1192     1756  4      If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  1193     1757  4          (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
;  1194     1758  3      Then
;  1195     1759  3          !
;  1196     1760  3          ! Determine if the device recorded by this volume
;  1197     1761  3          ! mount or dismount is a 'disk' type device.
;  1198     1762  3          !
;  1199     1763  4          Begin
;  1200     1764  4          If NOT TRANSLATE_CLASS (emb[emb$t_vm_namtxt],DC$_DISK)
;  1201     1765  4          Then
;  1202     1766  4              !
;  1203     1767  4              ! Indicate that the device recorded by this entry is
;  1204     1768  4              ! not a 'disk', by returning to the calling routine
;  1205     1769  4              ! with a false value.
;  1206     1770  4              !
;  1207     1771  4              Return false
;  1208     1772  4          Else
;  1209     1773  4              Return true ;
;  1210     1774  3          End ;
;  1211     1775  3
;  1212     1776  5      If ( ((.emb[emb$w_hd_entry] EQLU EMB$K_LM) AND
;  1213     1777  4              (.emb[emb$b_lm_class] EQLU DC$_DISK)) OR
;  1214     1778  4
;  1215     1779  5          ((.emb[emb$w_hd_entry] EQLU EMB$K_SP) AND
;  1216     1780  4              (.emb[emb$b_sp_class] EQLU DC$_DISK)) OR
;  1217     1781  4
;  1218     1782  4      ! Entry type must be either a device error, timeout, or attention.
;  1219     1783  4      !
;  1220     1784  4          (.emb[emb$b_dv_class] EQLU DC$_DISK) )
;  1221     1785  3      Then
;  1222     1786  3          !
;  1223     1787  3          ! Indicate that this entry does match a selected
;  1224     1788  3          ! device class, by returning to the calling routine
;  1225     1789  3          ! with a true value.
;  1226     1790  3          !
;  1227     1791  3          Return true ;
;  1228     1792  3
;  1229     1793  3      !
;  1230     1794  3      ! Determine whether this is disk related unsolicited mscp entry.
;  1231     1795  3      !
```

```
;  1232     1796  3     If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
;  1233     1797  3         CH$EQL (2,emb[driver_type],2,CH$PTR(uplit('DISK')))
;  1234     1798  3     Then
;  1235     1799  3         ! Yes, return to the calling routine with a true value.
;  1236     1800  3         !
;  1237     1801  3         Return true ;
;  1238     1802  2     End ;
;  1239     1803  2
;  1240     1804  2 !
;  1241     1805  2 ! Determine if 'REALTIME' entries are selected.
;  1242     1806  2 !
;  1243     1807  2 If ((.exclude_mask[exc$v_realtime]) OR
;  1244     1808  3     (.include_mask[inc$v_realtime]))
;  1245     1809  2 Then
;  1246     1810  2     !
;  1247     1811  2     ! Determine if the device recorded by this entry, matches the
;  1248     1812  2     ! selected device class.
;  1249     1813  2     !
;  1250     1814  3     Begin
;  1251     1815  3     If .emb[emb$b_dv_class] EQLU DC$_REALTIME
;  1252     1816  3     Then
;  1253     1817  3         !
;  1254     1818  3         ! Indicate that this entry does match a selected
;  1255     1819  3         ! device class, by returning to the calling routine
;  1256     1820  3         ! with a true value.
;  1257     1821  3         !
;  1258     1822  3         Return true ;
;  1259     1823  2     End ;
;  1260     1824  2
;  1261     1825  2 !
;  1262     1826  2 ! Determine if 'SYNCHRONOUS COMMUNICATION' entries are selected.
;  1263     1827  2 !
;  1264     1828  3 If ((.exclude_mask[exc$v_sync_comm]) OR
;  1265     1829  3     (.include_mask[inc$v_sync_comm]))
;  1266     1830  2 Then
;  1267     1831  2     !
;  1268     1832  2     ! Determine if the device recorded by this entry, matches the
;  1269     1833  2     ! selected device class.
;  1270     1834  2     !
;  1271     1835  3     Begin
;  1272     1836  3     If .emb[emb$b_dv_class] EQLU DC$_SCOM
;  1273     1837  3     Then
;  1274     1838  3         !
;  1275     1839  3         ! Indicate that this entry does match a selected
;  1276     1840  3         ! device class, by returning to the calling routine
;  1277     1841  3         ! with a true value.
;  1278     1842  3         !
;  1279     1843  3         Return true ;
;  1280     1844  2     End ;
;  1281     1845  2
;  1282     1846  2 !
;  1283     1847  2 ! Determine if 'TAPE' entries are selected.
;  1284     1848  2 !
;  1285     1849  3 If ((.exclude_mask[exc$v_tapes]) OR
;  1286     1850  3     (.include_mask[inc$v_tapes]))
;  1287     1851  2 Then
;  1288     1852  2     !
```

```
;  1289     1853  2      ! Determine if the device recorded by this entry, matches the
;  1290     1854  2      ! selected device class.
;  1291     1855  2      !
;  1292     1856  3      Begin
;  1293     1857  4      If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  1294     1858  4          (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
;  1295     1859  3      Then
;  1296     1860  3          !
;  1297     1861  3          ! Determine if the device recorded by this volume
;  1298     1862  3          ! mount or dismount is a 'tape' type device.
;  1299     1863  3          !
;  1300     1864  4          Begin
;  1301     1865  4          If NOT TRANSLATE_CLASS (emb[emb$t_vm_namtxt],DC$_TAPE)
;  1302     1866  4          Then
;  1303     1867  4              !
;  1304     1868  4              ! Indicate that the device recorded by this entry is
;  1305     1869  4              ! not a 'tape', by returning to the calling routine
;  1306     1870  4              ! with a false value.
;  1307     1871  4              !
;  1308     1872  4              Return false
;  1309     1873  4          Else
;  1310     1874  4              Return true ;
;  1311     1875  3          End ;
;  1312     1876  3
;  1313     1877  5      If ( ((.emb[emb$w_hd_entry] EQLU EMB$K_LM) AND
;  1314     1878  4              (.emb[emb$b_lm_class] EQLU DC$_TAPE)) OR
;  1315     1879  4
;  1316     1880  5          ((.emb[emb$w_hd_entry] EQLU EMB$K_SP) AND
;  1317     1881  4              (.emb[emb$b_sp_class] EQLU DC$_TAPE)) OR
;  1318     1882  4
;  1319     1883  4      ! Entry type must be either a device error, timeout, or attention.
;  1320     1884  4      !
;  1321     1885  4          (.emb[emb$b_dv_class] EQLU DC$_TAPE) )
;  1322     1886  3      Then
;  1323     1887  3          !
;  1324     1888  3          ! Indicate that this entry does match a selected
;  1325     1889  3          ! device class, by returning to the calling routine
;  1326     1890  3          ! with a true value.
;  1327     1891  3          !
;  1328     1892  3          Return true ;
;  1329     1893  3
;  1330     1894  3      !
;  1331     1895  3      ! Determine whether this is tape related unsolicited mscp entry.
;  1332     1896  3      !
;  1333     1897  3      If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
;  1334     1898  3          CH$EQL (2,emb[driver_type],2,CH$PTR(uplit('TAPE')))
;  1335     1899  3      Then
;  1336     1900  3          ! Yes, return to the calling routine with a true value.
;  1337     1901  3          !
;  1338     1902  3          Return true ;
;  1339     1903  2      End ;
;  1340     1904  2
;  1341     1905  2  !
;  1342     1906  2  ! Determine if 'MISC' entries are selected.
;  1343     1907  2  !
;  1344     1908  2  !If ((.exclude_mask[exc$v_misc]) OR
;  1345     1909  2  !    (.include_mask[inc$v_misc]))
```

```
;  1346     1910  2 !Then
;  1347     1911  2     !
;  1348     1912  2     ! Determine if the device recorded by this entry, matches the
;  1349     1913  2     ! selected device class.
;  1350     1914  2     !
;  1351     1915  2 !    Begin
;  1352     1916  2 !    If .emb[emb$b_dv_class] EQLU DC$_MISC
;  1353     1917  2 !    Then
;  1354     1918  2         !
;  1355     1919  2         ! Indicate that this entry does match a selected
;  1356     1920  2         ! device class, by returning to the calling routine
;  1357     1921  2         ! with a true value.
;  1358     1922  2         !
;  1359     1923  2 !       Return true ;
;  1360     1924  2 !    End ;
;  1361     1925  2
;  1362     1926  2 !
;  1363     1927  2 ! Determine if 'LP' entries are selected.
;  1364     1928  2 !
;  1365     1929  2 !If ((.exclude_mask[exc$v_line_printr]) OR
;  1366     1930  2 !    (.include_mask[inc$v_line_printr]))
;  1367     1931  2 !Then
;  1368     1932  2     !
;  1369     1933  2     ! Determine if the device recorded by this entry, matches the
;  1370     1934  2     ! selected device class.
;  1371     1935  2     !
;  1372     1936  2 !    Begin
;  1373     1937  2 !    If .emb[emb$b_dv_class] EQLU DC$_LP
;  1374     1938  2 !    Then
;  1375     1939  2         !
;  1376     1940  2         ! Indicate that this entry does match a selected
;  1377     1941  2         ! device class, by returning to the calling routine
;  1378     1942  2         ! with a true value.
;  1379     1943  2         !
;  1380     1944  2 !       Return true ;
;  1381     1945  2 !    End ;
;  1382     1946  2
;  1383     1947  2 !
;  1384     1948  2 ! Determine if 'JOURNAL' entries are selected.
;  1385     1949  2 !
;  1386     1950  2 !If ((.exclude_mask[exc$v_journal]) OR
;  1387     1951  2 !    (.include_mask[inc$v_journal]))
;  1388     1952  2 !Then
;  1389     1953  2     !
;  1390     1954  2     ! Determine if the device recorded by this entry, matches the
;  1391     1955  2     ! selected device class.
;  1392     1956  2     !
;  1393     1957  2 !    Begin
;  1394     1958  2 !    If .emb[emb$b_dv_class] EQLU DC$_JOURNAL
;  1395     1959  2 !    Then
;  1396     1960  2         !
;  1397     1961  2         ! Indicate that this entry does match a selected
;  1398     1962  2         ! device class, by returning to the calling routine
;  1399     1963  2         ! with a true value.
;  1400     1964  2         !
;  1401     1965  2 !       Return true ;
;  1402     1966  2 !    End ;
```

```
;  1403          1967  2
;  1404          1968  2 !
;  1405          1969  2 ! Indicate that this entry does not match any of the selected
;  1406          1970  2 ! device classes, by returning to the calling routine with a
;  1407          1971  2 ! false value.
;  1408          1972  2 !
;  1409          1973  2 Return false ;
;  1410          1974  1 End ;   ! Routine


                                                        .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2

                                    4B 53 49 44  00000 P.AAA:  .ASCII  \DISK\                                    ;
                                    45 50 41 54  00004 P.AAB:  .ASCII  \TAPE\                                    ;


                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                               003C 00000 VERIFY_DEVICE_CLASS:
                                                        .WORD   Save R2,R3,R4,R5                                 ; 1682
                                 55 00000000G   00  9E 00002        MOVAB   EXCLUDE_MASK, R5                     ;
                                 54 00000000G   00  9E 00009        MOVAB   INCLUDE_MASK, R4                     ;
                                 53 00000000G   00  9E 00010        MOVAB   EMB+16, R3                           ;
                                 52       F4    A3  3C 00017        MOVZWL  EMB+4, R2                            ; 1712
                           0065  8F             52  B1 0001B        CMPW    R2, #101                             ;
                                                11  12 00020        BNEQ    1$                                   ;
                                 50             64  D0 00022        MOVL    INCLUDE_MASK, R0                     ; 1713
                             0A  60             02  E0 00025        BBS     #2, (R0), 1$                         ;
                                 50             64  D0 00029        MOVL    INCLUDE_MASK, R0                     ; 1714
                                 03       01    A0  E8 0002C        BLBS    1(R0), 7$                            ;
                                             010A  31 00030        BRW     28$                                  ;
                                 51             65  D0 00033 1$:    MOVL    EXCLUDE_MASK, R1                     ; 1721
                             07  61             01  E0 00036        BBS     #1, (R1), 2$                         ;
                                 50             64  D0 0003A        MOVL    INCLUDE_MASK, R0                     ; 1722
                             21  60             01  E1 0003D        BBC     #1, (R0), 5$                         ;
                           0064  8F             52  B1 00041 2$:    CMPW    R2, #100                             ; 1729
                                                06  12 00046        BNEQ    3$                                   ;
                             80  8F             63  91 00048        CMPB    EMB+16, #128                         ; 1730
                                                77  13 0004C        BEQL    13$                                  ;
                           0063  8F             52  B1 0004E 3$:    CMPW    R2, #99                              ; 1732
                                                06  12 00053        BNEQ    4$                                   ;
                             80  8F             63  91 00055        CMPB    EMB+16, #128                         ; 1733
                                                7B  13 00059        BEQL    16$                                  ;
                             80  8F       0C    A3  91 0005B 4$:    CMPB    EMB+28, #128                         ; 1735
                                                74  13 00060        BEQL    16$                                  ;
                             07  61             02  E0 00062 5$:    BBS     #2, (R1), 6$                         ; 1748
                                 50             64  D0 00066        MOVL    INCLUDE_MASK, R0                     ; 1749
                             45  60             02  E1 00069        BBC     #2, (R0), 11$                        ;
                           0040  8F             52  B1 0006D 6$:    CMPW    R2, #64                              ; 1756
                                                07  13 00072        BEQL    7$                                   ;
                           0041  8F             52  B1 00074        CMPW    R2, #65                              ; 1757
                                                04  12 00079        BNEQ    8$                                   ;
                                                01  DD 0007B 7$:    PUSHL   #1                                   ; 1764
                                                78  11 0007D        BRB     20$                                  ;
                                 50       F4    A3  3C 0007F 8$:    MOVZWL  EMB+4, R0                            ; 1776
```

```
                        0064    8F              50  B1 00083          CMPW    R0, #100                        ;
                                                05  12 00088          BNEQ    9$                              ;
                                01              63  91 0008A          CMPB    EMB+16, #1                      ; 1777
                                                47  13 0008D          BEQL    16$                             ;
                        0063    8F              50  B1 0008F 9$:      CMPW    R0, #99                         ; 1779
                                                05  12 00094          BNEQ    10$                             ;
                                01              63  91 00096          CMPB    EMB+16, #1                      ; 1780
                                                79  13 00099          BEQL    22$                             ;
                                01      0C      A3  91 0009B 10$:     CMPB    EMB+28, #1                      ; 1784
                                                7F  13 0009F          BEQL    24$                             ;
                        0065    8F              50  B1 000A1          CMPW    R0, #101                        ; 1796
                                                0A  12 000A6          BNEQ    11$                             ;
                    00000000'   00      02      A3  B1 000A8          CMPW    EMB+18, P.AAA                   ; 1797
                                                74  13 000B0          BEQL    26$                             ;
                                51              65  D0 000B2 11$:     MOVL    EXCLUDE_MASK, R1                ; 1807
                07              61              06  E0 000B5          BBS     #6, (R1), 12$                   ;
                                50              64  D0 000B9          MOVL    INCLUDE_MASK, R0                ; 1808
                07              60              06  E1 000BC          BBC     #6, (R0), 14$                   ;
                          60    8F      0C      A3  91 000C0 12$:     CMPB    EMB+28, #96                     ; 1815
                                                72  13 000C5 13$:     BEQL    27$                             ;
                                                61  95 000C7 14$:     TSTB    (R1)                            ; 1828
                                                07  19 000C9          BLSS    15$                             ;
                                50              64  D0 000CB          MOVL    INCLUDE_MASK, R0                ; 1829
                                                60  95 000CE          TSTB    (R0)                            ;
                                                06  18 000D0          BGEQ    17$                             ;
                                20      0C      A3  91 000D2 15$:     CMPB    EMB+28, #32                     ; 1836
                                                61  13 000D6 16$:     BEQL    27$                             ;
                                07      01      A1  E8 000D8 17$:     BLBS    1(R1), 18$                      ; 1849
                                50              64  D0 000DC          MOVL    INCLUDE_MASK, R0                ; 1850
                                5A      01      A0  E9 000DF          BLBC    1(R0), 28$                      ;
                                50      F4      A3  3C 000E3 18$:     MOVZWL  EMB+4, R0                       ; 1857
                        0040    8F              50  B1 000E7          CMPW    R0, #64                         ;
                                                07  13 000EC          BEQL    19$                             ;
                        0041    8F              50  B1 000EE          CMPW    R0, #65                         ; 1858
                                                11  12 000F3          BNEQ    21$                             ;
                                                02  DD 000F5 19$:     PUSHL   #2                              ; 1865
                                        0F      A3  9F 000F7 20$:     PUSHAB  EMB+31                          ;
                    00000000V   00              02  FB 000FA          CALLS   #2, TRANSLATE_CLASS             ;
                                35              50  E8 00101          BLBS    R0, 27$                         ;
                                                37  11 00104          BRB     28$                             ; 1874
                                50      F4      A3  3C 00106 21$:     MOVZWL  EMB+4, R0                       ; 1877
                        0064    8F              50  B1 0010A          CMPW    R0, #100                        ;
                                                05  12 0010F          BNEQ    23$                             ;
                                02              63  91 00111          CMPB    EMB+16, #2                      ; 1878
                                                23  13 00114 22$:     BEQL    27$                             ;
                        0063    8F              50  B1 00116 23$:     CMPW    R0, #99                         ; 1880
                                                05  12 0011B          BNEQ    25$                             ;
                                02              63  91 0011D          CMPB    EMB+16, #2                      ; 1881
                                                17  13 00120 24$:     BEQL    27$                             ;
                                02      0C      A3  91 00122 25$:     CMPB    EMB+28, #2                      ; 1885
                                                11  13 00126 26$:     BEQL    27$                             ;
                        0065    8F              50  B1 00128          CMPW    R0, #101                        ; 1897
                                                0E  12 0012D          BNEQ    28$                             ;
                    00000000'   00      02      A3  B1 0012F          CMPW    EMB+18, P.AAB                   ; 1898
                                                04  12 00137          BNEQ    28$                             ;
                                50              01  D0 00139 27$:     MOVL    #1, R0                          ; 1902
                                                    04 0013C          RET                                     ;
```

```
                50  D4 0013D 28$:    CLRL    R0                              ; 1974
                    04 0013F          RET                                     ;

; Routine Size:  320 bytes,    Routine Base:  $CODE + 04E1

; 1411          1975  1
```

```
;  1413     1976  1 Routine VERIFY_DEVICE =
;  1414     1977  2 Begin
;  1415     1978  2
;  1416     1979  2 !++
;  1417     1980  2 !
;  1418     1981  2 !--
;  1419     1982  2
;  1420     1983  2 Local
;  1421     1984  2     Dev_name,
;  1422     1985  2     Dev_name_length,
;  1423     1986  2     Dev_unit,
;  1424     1987  2     Status ;
;  1425     1988  2
;  1426     1989  2 Bind
;  1427     1990  2     lm_name_length = emb[emb$t_lm_devnam] : BYTE,
;  1428     1991  2     sp_name_length = emb[emb$t_sp_devnam] : BYTE,
;  1429     1992  2     dv_name_length = emb[emb$t_dv_name] : BYTE ;
;  1430     1993  2
;  1431     1994  2 !
;  1432     1995  2 ! Determine whether this is an unsolicited mscp entry and
;  1433     1996  2 ! return with a false value if so (logmscp entries are not
;  1434     1997  2 ! applicable to a specific device).
;  1435     1998  2 !
;  1436     1999  2 If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP
;  1437     2000  2 Then
;  1438     2001  2     Return false ;
;  1439     2002  2
;  1440     2003  2 !
;  1441     2004  2 ! Determine the type of entry so that the comparison for the
;  1442     2005  2 ! device class is made against the appropriate field in the entry.
;  1443     2006  2 !
;  1444     2007  2 ! Determine if this a log message entry.
;  1445     2008  2 !
;  1446     2009  2 If .emb[emb$w_hd_entry] EQLU EMB$K_LM
;  1447     2010  2 Then
;  1448     2011  2     !
;  1449     2012  2     ! Entry type is a log message, get the device name,
;  1450     2013  2     ! name length, and unit number.
;  1451     2014  2     !
;  1452     2015  3     Begin
;  1453     2016  3     Dev_name = emb[emb$t_lm_devnam] + 1 ;
;  1454     2017  3     Dev_name_length = .lm_name_length ;
;  1455     2018  3     Dev_unit = .emb[emb$w_lm_unit] ;
;  1456     2019  3     End
;  1457     2020  2 Else
;  1458     2021  2     !
;  1459     2022  2     ! Determine if this is a log status entry.
;  1460     2023  2     !
;  1461     2024  3     Begin
;  1462     2025  3     If .emb[emb$w_hd_entry] EQLU EMB$K_SP
;  1463     2026  3     Then
;  1464     2027  3         !
;  1465     2028  3         ! Entry type is a log status, get the device name,
;  1466     2029  3         ! name length, and unit number.
;  1467     2030  3         !
;  1468     2031  4         Begin
;  1469     2032  4         Dev_name = emb[emb$t_sp_devnam] + 1 ;
```

```
;  1470          2033  4          Dev_name_length = .sp_name_length ;
;  1471          2034  4          Dev_unit = .emb[emb$w_sp_unit] ;
;  1472          2035  4          End
;  1473          2036  3      Else
;  1474          2037  3          !
;  1475          2038  3          ! Determine if this a volume mount/dismount entry.
;  1476          2039  3          !
;  1477          2040  4          Begin
;  1478          2041  5          If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  1479          2042  5              (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
;  1480          2043  4          Then
;  1481          2044  4              !
;  1482          2045  4              ! Entry type is a either a volume mount/dismount, get
;  1483          2046  4              ! the device name, name length, and unit number.
;  1484          2047  4              !
;  1485          2048  5              Begin
;  1486          2049  5              Dev_name = emb[emb$t_vm_namtxt] ;
;  1487          2050  5              Dev_name_length = .emb[emb$b_vm_namlng] ;
;  1488          2051  5              Dev_unit = .emb[emb$w_vm_unit] ;
;  1489          2052  5              End
;  1490          2053  4          Else
;  1491          2054  4              !
;  1492          2055  4              ! Entry type must be either a device error, device timeount,
;  1493          2056  4              ! or a device attention, get the device name, name length, and
;  1494          2057  4              ! unit number.
;  1495          2058  4              !
;  1496          2059  5              Begin
;  1497          2060  5              Dev_name = emb[emb$t_dv_name] + 1 ;
;  1498          2061  5              Dev_name_length = .dv_name_length ;
;  1499          2062  5              Dev_unit = .emb[emb$w_dv_unit] ;
;  1500          2063  4              End ;
;  1501          2064  3          End ;
;  1502          2065  2      End ;
;  1503          2066  2
;  1504          2067  2 !
;  1505          2068  2 ! Call the search queue routine to determine if the device recorded by
;  1506          2069  2 ! this entry matches any of the selected devices.
;  1507          2070  2 !
;  1508          2071  2 Status = SEARCH_QUEUE (.dev_name,dev_name_length,dev_unit) ;
;  1509          2072  2
;  1510          2073  2 !
;  1511          2074  2 ! Return the status from the search queue operation to the
;  1512          2075  2 ! calling routine.
;  1513          2076  2 !
;  1514          2077  2 .Status
;  1515          2078  1 End ;   ! Routine


                                0004 00000 VERIFY_DEVICE:
                                           .WORD   Save R2                                  ; 1976
                    52 00000000G   00  9E 00002    MOVAB   EMB+4, R2                        ;
                    5E             08  C2 00009    SUBL2   #8, SP                           ;
                    50             62  3C 0000C    MOVZWL  EMB+4, R0                        ; 1999
              0065  8F             50  B1 0000F    CMPW    R0, #101                         ;
```

```
                                 03  12 00014          BNEQ    1$                                                      ;
                                 50  D4 00016          CLRL    R0                                                      ; 2001
                                 04  00018          RET                                                             ;
                          0064   8F      50  B1 00019 1$:    CMPW    R0, #100                                        ; 2009
                                 0F  12 0001E          BNEQ    2$                                                      ;
                                 51      11  A2 9E 00020      MOVAB   EMB+21, DEV_NAME                                ; 2016
                            04   AE      10  A2 9A 00024      MOVZBL  LM_NAME_LENGTH, DEV_NAME_LENGTH                 ; 2017
                                 6E      0E  A2 3C 00029      MOVZWL  EMB+18, DEV_UNIT                                ; 2018
                                 3C  11 0002D          BRB     7$                                                      ; 2009
                          0063   8F      50  B1 0002F 2$:    CMPW    R0, #99                                         ; 2025
                                 0B  12 00034          BNEQ    3$                                                      ;
                                 51      3D  A2 9E 00036      MOVAB   EMB+65, DEV_NAME                                ; 2032
                            04   AE      3C  A2 9A 0003A      MOVZBL  SP_NAME_LENGTH, DEV_NAME_LENGTH                 ; 2033
                                 26  11 0003F          BRB     6$                                                      ; 2034
                          0040   8F      50  B1 00041 3$:    CMPW    R0, #64                                         ; 2041
                                 07  13 00046          BEQL    4$                                                      ;
                          0041   8F      50  B1 00048        CMPW    R0, #65                                         ; 2042
                                 0F  12 0004D          BNEQ    5$                                                      ;
                                 51      1B  A2 9E 0004F 4$:    MOVAB   EMB+31, DEV_NAME                                ; 2049
                            04   AE      1A  A2 9A 00053      MOVZBL  EMB+30, DEV_NAME_LENGTH                         ; 2050
                                 6E      18  A2 3C 00058      MOVZWL  EMB+28, DEV_UNIT                                ; 2051
                                 0D  11 0005C          BRB     7$                                                      ; 2041
                                 51      3B  A2 9E 0005E 5$:    MOVAB   EMB+63, DEV_NAME                                ; 2060
                            04   AE      3A  A2 9A 00062      MOVZBL  DV_NAME_LENGTH, DEV_NAME_LENGTH                 ; 2061
                                 6E      26  A2 3C 00067 6$:    MOVZWL  EMB+42, DEV_UNIT                                ; 2062
                                 5E  DD 0006B 7$:    PUSHL   SP                                                      ; 2071
                                     08  AE 9F 0006D      PUSHAB  DEV_NAME_LENGTH                                     ;
                                 51  DD 00070          PUSHL   DEV_NAME                                                ;
                    00000000G    00      03  FB 00072      CALLS   #3, SEARCH_QUEUE                                    ;
                                 04  00079          RET                                                             ; 2078

; Routine Size:  122 bytes,    Routine Base:  $CODE + 0621

; 1516          2079  1
```

```
;  1518     2080  1 GLOBAL ROUTINE TRANSLATE_CLASS (search_name,dev_class) =
;  1519     2081  2 Begin
;  1520     2082  2
;  1521     2083  2 !++
;  1522     2084  2 !
;  1523     2085  2 ! Functional Description:
;  1524     2086  2 !
;  1525     2087  2 !    This routine searches the device tables to verify the device
;  1526     2088  2 !    class and device name.
;  1527     2089  2 !
;  1528     2090  2 ! Calling Sequence:
;  1529     2091  2 !
;  1530     2092  2 !    TRANSLATE_CLASS (search_name,dev_class)
;  1531     2093  2 !
;  1532     2094  2 ! Input Parameters:
;  1533     2095  2 !
;  1534     2096  2 !    Search name = First two characters of device name
;  1535     2097  2 !
;  1536     2098  2 !    Dev_class = Device class to search for.
;  1537     2099  2 !
;  1538     2100  2 !
;  1539     2101  2 !    If the device class is found, then the specified device name
;  1540     2102  2 !    is compared against the device names in the device specific table.
;  1541     2103  2 !    Returns true if both match.
;  1542     2104  2 !
;  1543     2105  2 !    Returns false if device class and/or device name doesn't match.
;  1544     2106  2 !    (This should eventually be caught and handled by the parse_devname
;  1545     2107  2 !    routine.)
;  1546     2108  2 !
;  1547     2109  2 !--
;  1548     2110  2
;  1549     2111  2 EXTERNAL
;  1550     2112  2     Dev_addrs_ptr:      REF VECTOR [,long],
;  1551     2113  2     Dev_class_ptr:      REF VECTOR [,word],
;  1552     2114  2     Max_classes:        REF VECTOR [,byte];
;  1553     2115  2
;  1554     2116  2 OWN
;  1555     2117  2     I:          BYTE Initial (1),       ! Device address pointer index
;  1556     2118  2     Max_classes_value:  BYTE ;
;  1557     2119  2
;  1558     2120  2 LOCAL
;  1559     2121  2     Dev_specific_tbl:   REF VECTOR [,word], ! Device specific table address
;  1560     2122  2     K:          Initial (0) ;           ! Device specific table index
;  1561     2123  2
;  1562     2124  2 BIND
;  1563     2125  2     Cs_name = CH$PTR (uplit('CS')) ;
;  1564     2126  2
;  1565     2127  2 !
;  1566     2128  2 ! Device class ptr is the address of a table that contains supported device
;  1567     2129  2 ! classes and pointers to the device class specific information tables.
;  1568     2130  2 !
;  1569     2131  2 ! The device class specific table contains the supported device names,
;  1570     2132  2 ! image name pointers (image that needs to get activated), and transfer
;  1571     2133  2 ! address pointers.
;  1572     2134  2 !
;  1573     2135  2 ! This routine locates the matching device class retrieves the device
;  1574     2136  2 ! specific pointer and matches the specified device name against those
```

```
;  1575     2137  2 ! in the device specific table.
;  1576     2138  2 !
;  1577     2139  2 ! Loop through all of the device class entries.
;  1578     2140  2 !
;  1579     2141  2 Max_classes_value = max_classes[0] ;
;  1580     2142  2
;  1581     2143  2 Incr I from 1 to .max_classes_value do
;  1582     2144  3     Begin
;  1583     2145  3     If .dev_class_ptr[.I] EQL .dev_class
;  1584     2146  3     Then
;  1585     2147  4         Begin
;  1586     2148  4         !
;  1587     2149  4         ! Get the address of a device class specific table.
;  1588     2150  4         !
;  1589     2151  4         Dev_specific_tbl = .dev_addrs_ptr[.I] ;
;  1590     2152  4
;  1591     2153  4         !
;  1592     2154  4         ! Initialize another index for the device class specific table so don't
;  1593     2155  4         ! lose the current position. Determine if the contents of the device
;  1594     2156  4         ! name field is valid OR whether the end of the device name entries
;  1595     2157  4         ! in the table has been reached.
;  1596     2158  4         !
;  1597     2159  4         K = 1 ;
;  1598     2160  4         Until (.K EQL .dev_specific_tbl[0]) do
;  1599     2161  5             Begin
;  1600     2162  5             !
;  1601     2163  5             ! Determine if the selected device name matches any of the
;  1602     2164  5             ! device names recorded in this table.
;  1603     2165  5             !
;  1604     2166  5             If CH$EQL (2, CH$PTR(.search_name), 2, CH$PTR(dev_specific_tbl[.K]))
;  1605     2167  5             Then
;  1606     2168  5                 !
;  1607     2169  5                 ! The device names match. Using the class dir table index,
;  1608     2170  5                 ! get the corresponding device class.
;  1609     2171  5                 !
;  1610     2172  5                 Return true ;
;  1611     2173  5
;  1612     2174  5             !
;  1613     2175  5             ! Update the device name pointer indices.
;  1614     2176  5             !
;  1615     2177  5             K = .K + 1 ;
;  1616     2178  4             End ;
;  1617     2179  3         End ;
;  1618     2180  2     End ;
;  1619     2181  2
;  1620     2182  2
;  1621     2183  2 !
;  1622     2184  2 ! The name for the console device 'CSA' is not included in the device name
;  1623     2185  2 ! tables contained in ERFLIB.TLB. It really is a second device name for
;  1624     2186  2 ! the RX device which is included in the device tables. There should be
;  1625     2187  2 ! a table that includes devices like these, however because there is only
;  1626     2188  2 ! one at this time, it is checked for explicitly.
;  1627     2189  2 !
;  1628     2190  2 If CH$EQL (2, CH$PTR(.search_name), 2, cs_name)
;  1629     2191  2 Then
;  1630     2192  2     !
;  1631     2193  2     ! This is a 'CS' entry, determine whether the 'CS' device class
```

```
;  1632          2194  2      ! matches the device class being searched for.
;  1633          2195  2      !
;  1634          2196  2      Begin
;  1635          2197  2      If .dev_class EQL DC$_DISK
;  1636          2198  2      Then
;  1637          2199  2          ! Indicate that the device class matches by returning with
;  1638          2200  2          ! a true value.
;  1639          2201  2          !
;  1640          2202  2          Return true ;
;  1641          2203  2      End ;
;  1642          2204  2
;  1643          2205  2
;  1644          2206  2  !
;  1645          2207  2  ! Could not locate a class for this device name.
;  1646          2208  2  !
;  1647          2209  2  Return false ;
;  1648          2210  2
;  1649          2211  1  End ;           ! Routine


                                                        .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2

                                    00  00  53  43  00008 P.AAC:  .ASCII  \CS\<0><0>                                ;

                                                        .PSECT  $OWN$,NOEXE,  PIC,2

                                            01  00036 I:        .BYTE   1                                          ;
                                                00037 MAX_CLASSES_VALUE:
                                                        .BLKB   1

                                                CS_NAME=            P.AAC
                                                        .EXTRN  DEV_ADDRS_PTR, DEV_CLASS_PTR
                                                        .EXTRN  MAX_CLASSES

                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                              003C 00000        .ENTRY  TRANSLATE_CLASS, Save R2,R3,R4,R5          ; 2080
                                    55 00000000'   00   9E 00002        MOVAB   MAX_CLASSES_VALUE, R5              ;
                                                   52   D4 00009        CLRL    K                                  ; 2081
                                    65 00000000G   00   90 0000B        MOVB    MAX_CLASSES, MAX_CLASSES_VALUE     ; 2141
                                    54             65   9A 00012        MOVZBL  MAX_CLASSES_VALUE, R4              ; 2143
                                                   50   D4 00015        CLRL    I                                  ; 2145
                                                   32   11 00017        BRB     3$                                 ;
                                    51 00000000G   00   D0 00019 1$:     MOVL    DEV_CLASS_PTR, R1                  ;
                                                 6140   3F 00020        PUSHAW  (R1)[I]                            ;
08  AC              9E              10             00   ED 00023        CMPZV   #0, #16, @(SP)+, DEV_CLASS         ;
                                                   20   12 00029        BNEQ    3$                                 ;
                                    51 00000000G   00   D0 0002B        MOVL    DEV_ADDRS_PTR, R1                  ; 2151
                                    53             61   D0 00032        MOVL    (R1)[I], DEV_SPECIFIC_TBL          ;
                                    52             01   D0 00036        MOVL    #1, K                              ; 2159
    52              63              10             00   ED 00039 2$:     CMPZV   #0, #16, (DEV_SPECIFIC_TBL), K     ; 2160
                                                   0B   13 0003E        BEQL    3$                                 ;
                                  6342       04    BC   B1 00040        CMPW    @SEARCH_NAME, (DEV_SPECIFIC_TBL)[K]; 2166
                                                   18   13 00045        BEQL    4$                                 ;
                                                   52   D6 00047        INCL    K                                  ; 2177
                                                   EE   11 00049        BRB     2$                                 ; 2160
```

```
                    CA                  50              54  F3 0004B 3$:      AOBLEQ  R4, I, 1$                              ; 2143
                       00000000'        00        04    BC  B1 0004F        CMPW    @SEARCH_NAME, CS_NAME                   ; 2190
                                                    0A    12 00057          BNEQ    5$                                      ;
                                        01        08    AC  D1 00059        CMPL    DEV_CLASS, #1                           ; 2197
                                                    04    12 0005D          BNEQ    5$                                      ;
                                        50              01  D0 0005F 4$:      MOVL    #1, R0                                  ; 2202
                                                        04  00062           RET                                             ;
                                                    50  D4 00063 5$:        CLRL    R0                                      ; 2209
                                                        04  00065           RET                                             ; 2211

; Routine Size:  102 bytes,    Routine Base:  $CODE + 069B

; 1650          2212  1
; 1651          2213  1
; 1652          2214  1 End
; 1653          2215  0 ELUDOM



                                                                    .EXTRN  LIB$SIGNAL

;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $OWN$                          56  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $CODE                        1793  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $PLIT                          12  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                                   Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        72         0        1000        00:01.9



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RECSELECT/OBJ=OBJ$:RECSELECT MSRC$:RECSELECT/UPDATE=(ENH$:RECSELECT)

; Size:          1793 code + 68 data bytes
; Run Time:        00:40.9
; Elapsed Time:    01:21.8
; Lines/CPU Min:    3251
; Lexemes/CPU-Min: 19926
; Memory Used:  349 pages
```

```
; Compilation Complete
```

0153 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RECSELECT
LIS

RLDISK
LIS

PUDRIVER
LIS

PCL11T
LIS

ROLLUP
LIS

RXDISK
LIS

PCL11R
LIS

SB11
LIS

RKDISK
LIS